ABSTRACT

Title of dissertation: A PUBLIC HEALTH MODELING BASED APPROACH TO INFORMATION SECURITY QUANTIFICATION

Edward M. Condon, Doctor of Philosophy, 2015

Dissertation directed by: Professor Michel Cukier
Reliability Engineering Program

Modeling the occurrence of computer security incidents within a defined population of computers can be used to help understand some of factors contributing to risk and transmission of these incidents among the population. A better understanding of these factors can be used to determine appropriate intervention actions that can be applied to the population, which may also be evaluated through the application of models. Explanatory models attempt to include and account for various primary factors that affect the occurrence of computer security incidents.

Models based on observed security incidents may also be used to evaluate interventions even when explanatory models may not exist or may be difficult to formulate or express for a particular incident type. Forecasting models can be used to project the occurrence of incidents in the future and these projections can be compared to actual observations before and after interventions are applied.

The following aspects of modeling computer security incidents are explored: (1) the presentation and discussion of adapting some commonly used infectious disease

models for modeling the spread of some types of computer security incidents along with applicable intervention actions; (2) an illustration of how these types of models could be applied to making resource allocation decisions regarding intervention efforts; (3) the presentation and comparison of models that can be used for tracking/forecasting security incidents and monitoring the impact of interventions; (4) the presentation of a method for estimating model features and parameter distributions from observed data; and (5) the exploration of some population characteristics and models for evaluating where to focus or target intervention actions.

When resources for responding to or preventing computer security incidents are limited or constrained, the ability to accurately model and evaluate intervention actions can be a useful tool for making the most of these resources.

A PUBLIC HEALTH MODELING BASED APPROACH TO INFORMATION SECURITY QUANTIFICATION

by

Edward M. Condon

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2015

Advisory Committee:
Professor Michel Cukier, Chair/Advisor
Professor Rance Cleaveland, Dean's Representative
Professor Jennifer Golbeck
Professor Jeffrey Herrmann
Professor Ali Mosleh
Mr. Gerry Sneeringer, Special Member

Dedication

To the memory of Joseph Daly Katsouros, a colleague and friend.

Acknowledgments

I am grateful to all the people who have made this dissertation possible and who have contributed and been a part in shaping the many facets of my graduate experience.

First and foremost I'd like to thank my advisor, Professor Michel Cukier for providing me an invaluable opportunity to work on challenging and interesting projects over the past several years. His strong mentorship has been a key element in this effort.

I would also like to thank my doctoral committee: Dr. Rance Cleaveland, Dr. Jennifer Golbeck, Dr. Jeffrey Herrmann, Dr. Ali Mosleh, and Mr. Gerry Sneeringer, for sparing their invaluable time and providing valuable feedback.

Other former and current graduate students who have provided support and guidance, including Robin Berthier, Danielle Chrun, and Bertrand Sobesto. I would also like to thank Matthew Virgo for his insight and moral support throughout the process and to especially acknowledge the help and support from Dorothea Brosius with the document preparation in LaTeX.

I owe my deepest thanks to my family - my parents who have always inspired and supported me through many endeavors (including this one), and to my partner, Sveta, and my children, Ian and Michael, whose patience and tolerance have been a crucial component.

There are many others who have helped and assisted in many ways over the years and I thank you.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction and Background

1.1 Overview

Computer and network security incidents have financial consequences both for individuals and for organizations. Compromised computer resources often end up being used as part of financially motivated attacks against others [1]. Examples include large-scale botnets [2, 3] used for organized criminal activities [4] such as extortion and phishing/pharming [5] attacks, as well as distribution of pop-up advertisements, spam [1] and "scareware" [6]. However, other attacks may not be motivated by financial gains for the attackers but do result in financial losses for the victim. Examples from recent years of such attacks include those against HBGary Federal [7] and Sony Playstation Network [8]. Laws and regulations may also require companies to publicly disclose data breaches. This can negatively affect stock prices and other valuations of the organization [9, 10].

In addition to the risks posed by security incidents (financial fraud, data breaches, reputational damage), there are direct costs to organizations for responding to such incidents. Organizations spend time and money developing and enforcing policies designed to reduce the number or severity of incidents. For example, corporations may purchase intrusion detection systems (IDS) and intrusion prevention

systems (IPS) to assist with monitoring and detecting incidents. Also, staff and user resources are needed to fix and recover from incidents. Most importantly, critical information technology infrastructure resources may be unavailable for use during the recovery process. Identifying and responding to incidents quickly can help minimize the impact within an organization, as well as to external entities that may be targeted by the use of an organization's resources (such as with reflector based denial-of-service attacks).

Undetected and/or unaddressed security incidents can have more severe consequences over time:

- Providing entry points to more critical systems or protected information;
- Being used as launching points for other attacks against the same or other organizations;
- Being used to gather and collect information about an organization that can be leveraged in the future (in the form of more targeted attacks or business competition);
- Causing reputational damage if regulatory requirements mandate the disclosure of applicable incidents or resulting breaches.

A recent example of a large scale breach illustrates some of the costs involved. The South Carolina Department of Revenue was involved in a breach where data involving 6.4 million consumers and businesses were stolen. One source [11] reports the cost as $700,000 for the hiring of a security firm to review what occurred and

make security suggestions. The same source reports that a similar security assessment before the breach would have cost about $200,000, which shows some of the additional costs resulting from the breach. In addition to the security assessment, the source reports there is a $12 million contract with Experian to provide a year of free credit-report monitoring to taxpayers; additional likely costs include hiring a PR firm and outside lawyers.

In many cases this kind of breach is the result of several security failings that have been combined or have cascaded into the detected and publicized or reported breach. Smaller computer security incidents can often form the building blocks of larger scale security breaches. Verizon reports that in 2011, 98% of breaches stemmed from external agents and that 97% of breaches were avoidable through simple or intermediate controls [12].

Sometimes the root causes of incidents may be well known and/or understood. However, sometimes the root causes will never be determined due to time, technical, and financial constraints. Even in cases where the root causes of incidents may be well known or understood, determining mitigation strategies and measuring the effectiveness of their implementation can be difficult. In the same way that understanding the mechanics behind a disease's effects on an organism does not always directly lead to a prevention or cure (generating an effective immune system response can still be a problem), determining effective responses to computer security threats is not always straightforward. Also, even when effective intervention actions are known, their timely implementation and deployment can still present additional challenges.

As concerns increase about the impact of such breaches, calls for changes in laws and policies to improve cyber security have also increased. Some laws and policies (such as the Computer Fraud and Abuse Act of 1986) outline activities subject to penalties and fines to discourage certain types of activity. Other laws and policies (such as the Health Information Technology for Economic and Clinical Health Act of 2009) impose penalties against organizations or otherwise provide incentives for these organizations to implement steps that may improve their resilience against some of the attacks and scenarios that have led to large breaches in the past. However, legislative measures can become outdated as technology and its use continue to evolve. Threats and attacks will often evolve in response to new uses of technology and to new defensive measures, similar to the way in which last year's flu vaccine may not be effective against this year's influenza [13].

1.2 Statement of Problem

Computer security personnel need to be able both to prepare and to react to incoming data from multiple sources in order to make assessments of changes in the state of "health" of their organization's information infrastructure, resources, and assets, and to decide upon actions or interventions. These personnel also need to have information regarding the types of appropriate interventions available, the likely effectiveness of such interventions, and the means to measure the impact–even when the root causes of a growing issue are not well known or understood. The identification and quantification of risk factors can assist with the development and

targeting of effective intervention and prevention strategies and models can play an important role both in identifying potential risk factors and in evaluating intervention and prevention implementations.

This dissertation addresses these key questions:

1. What are some of the available models that can be applied to computer security incident data, and how are they applicable?

2. Are some models better suited to some incident types based on features or properties of the incident type?

3. Can current models be adapted to better include relevant characteristics and dynamics for some incident types?

4. Can some models be used to explore the use of different interventions relevant to some incident types for planning purposes?

5. Are there properties or features of a computer population that may be useful for guiding and targeting interventions to reduce the occurrence of computer security incidents?

1.3 Outline of Dissertation

This dissertation is organized into eight chapters. Chapter 1 introduces the problem and relevant background. Chapter 2 provides a review of current literature which covers current techniques in the area of modeling relevant to computer security incidents. Chapters 3, 4, and 5 review and present the types of models applied to

computer security incidents in this study, along with some illustrative examples and simulations. Chapter 3 introduces models originally developed for modeling the spread of infectious diseases in animals and humans. Chapter 4 discusses adaptation of the infectious disease models to apply to incidents that spread via email. Chapter 5 examine two different types of quality control based models. Chapter 6 describes a real world computer security data set and illustrates the application of models using this data set. Chapter 7 discusses and illustrates how including additional factors to the data could be applied to intervention efforts. Chapter 8 summarizes the findings and provides recommendations for additional areas of study.

1.4 Background

1.4.1 Taxonomy/definition/description of computer security and incidents

Some definitions of computer security include:

- Information security is defined as the preservation of confidentiality, integrity and availability of information; in addition, other properties such as authenticity, accountability, non-repudiation and reliability can be involved [14].
- Information system security is a system characteristic and a set of mechanisms that span the system both logically and physically. The five security goals are integrity, availability, confidentiality, accountability, and assurance [15].
- A computer is secure if you can depend on it and its software to behave as you

expect [16].

It is easier to define and track instances of computer security failures than it is to comprehensively define and assess the overall computer security "health" of an organization. Conducting internal security audits is one method organizations can use to verify that security measures are in place [17]. However, security audits do not necessarily provide guidance or direct information regarding the effectiveness of security measures. Additional information is needed for organizations to make informed decisions regarding security measures.

Within organizations, computer security failings can often be simplified to a failing of a component or several components of a simple system involving a computer and its connections (network), human users, and often an external attacker.

- Computer/network–"hosts" and the communications/contacts between them. Physical computer networks may have a variety of topologies overlaid onto them–communications infrastructure (servers/clients), social contacts (email, instant messaging, social media apps). The computer itself usually has several subcomponents that can be part of a security failure–such as the operating system, software applications, and some network services.

- External (attackers)–source of attempts to gain unauthorized access to resources. Attacks can be active or passive (e.g., requiring some type of user action to occur before executing), remote or local (which may result in privilege escalation), and can target computers/networks as well as users.

- Human user(s)–interact(s) with local and remote computers, evaluates informa-

tion, can be responsible for generating some of the observable computer/network activity related to a computer.

The definition of what constitutes a computer or network security incident can be very context dependent. Howard and Longstaff [18] distinguish an "event" as the discrete change in state or status of a system as the result of a directed action towards a target, where a target can be a specific computer, account, process, or other logical or physical entity. An event may be part of normal operating and authorized activities, or it may be part of an attack, which is defined by Howard and Longstaff as "a series of steps taken by an attacker to achieve an unauthorized result" [18]. An incident is later defined as "a group of attacks that can be distinguished from other attacks because of the distinctiveness of the attackers, attacks, objectives, sites, and timing." Hansman and Hunt [19] give a similar but simpler definition of an incident as "an attempt at violating security policy, such as attacking a computer or attempting to gain unauthorised access to some data." A common key distinction made by various definitions is that an incident involves an aspect of unauthorized action or access to some resource (network, data, other).

1.4.2 Purposes of modeling

Models are conceptual tools to explain how an object or system of objects will behave. They range from simple to complex and usually involve trade-offs among accuracy, transparency, and flexibility [20]. Models can have two distinct roles–prediction and understanding. These roles are not necessarily independent nor

in opposition to each other, but models may be employed to serve one role more than the other. Models can also provide a means to isolate factors in a system and examine their role in situations where they could be difficult to examine independently.

Organizations need to be able to recognize and quantify potential problems quickly so that decisions can be made regarding preventative or corrective actions. Organizations want to know if a particular problem is expected to get worse or if any previous measures have affected the expected trajectory of an issue. Using models to fit existing computer security incident data allows organizations to better assess expected levels of incident occurrence as well as to compare expected levels of incident occurrence with the number of actual occurrences following the use of corrective measures.

Applicable models enable researchers to identify characteristics of incident occurrence over time and quantify current trends to provide insight into expected future trends. Predictive models developed from the analysis of security incident data can be used by organizations to decide if corrective or preventative actions are needed and to gauge or measure the impact of any changes made due to security policies and practices. This can be accomplished by comparing model forecasts to observed activity. If the observed number of incidents begins to deviate as an increasing trend away from the forecast values, an external factor may have changed (such as the development of a new variant of an attack), and an organization should reevaluate its control measures to make necessary adjustments.

A similar strategy can be used for evaluating the impact of changes made to the internal security environment (such as policy changes, IDS/IPS deployment,

system updates, user-awareness campaigns, etc.). Model forecasts of future incident levels can be compared to actual observed levels before and after implementing a new control measure. The effectiveness of the change to the security environment can be judged by comparing how well the observed number of incident occurrences tracks the forecast values for the time period after the new control measure has been implemented. Model forecasts for the use of different intervention types can also be compared to determine which available option is most appropriate based on an organization's goals.

Most of the models presented involve the number of computer security incident occurrences detected and confirmed. While each type of incident may have a different impact on an organization's overall security, counting the number of incidents provides an accessible and quantifiable measure related to overall security. This is similar to how counting the number of detected flaws in software code is used as a way to measure the quality of the software development process, even though simple counts may not differentiate between the types or severity of the detected flaws. An essential question is how to quantify security (or software quality) and some answers are implicit in the decisions made regarding the definition/selection of types of security incidents or software flaws to track or count.

1.4.3 Types of models

As previously mentioned, models are often used for two purposes–prediction and understanding. These roles may overlap and are not mutually exclusive. Models used

mostly for forecasting or prediction may be part of a quality control or monitoring type process. The models themselves may not shed much light on underlying dynamics nor explicitly suggest applicable interventions, but can be used to compare the impact of an applied corrective measure. Models used for understanding typically include aspects meant to capture some of the underlying dynamics and can also be used to explore potential outcomes based on implementing certain actions incorporated as part of the model.

1.4.3.1 Underlying dynamics models

Since many computer security incidents may be the result of a computer virus or worm and spread amongst computers, techniques used in the field of epidemiology and infectious disease surveillance can provide a useful framework. One approach to modeling infectious diseases in human and animal populations considers the state transitions involved with infectious disease transmission and recovery. The Susceptible-Infected-Recovered (SIR) model and its variants are an example of this approach. These models involve sets of differential equations to relate transitions between states and can include parameters for transmission and recovery rates.

More complex models may also factor in birth and death rates of a population as well as the proportion of the population vaccinated against a particular disease. Birth and death rates would be analogous to the rate at which new machines are added and the rate at which old machines are decommissioned from a population of computers. Vaccination would be similar to the installing a software update or

"patch" that fixes a particular vulnerability. Examples of using epidemic models to model computer worms and viruses include [21] and [22].

1.4.3.2 Quality control models

Two other types of models presented use historical data of incident occurrence to provide forecasts of future incident levels in time. These two types of models focus more on forecasting future levels than on accurately capturing underlying dynamics or causes. We will apply two types of these models to different types of computer security incidents to see which type of model may work best for different types of incidents and to see if there are some general observations that can be made about modeling the frequency of occurrence of computer security incidents.

When the symptoms or signs of a new type or strain of disease begin to affect human populations at noticeable and recorded levels, the underlying cause(s) may take time to determine. At this stage, often the frequency of symptom or disease occurrence is the only information available to make projections regarding future trends. The same is true with regard to computer security incidents. While the causes of some types of computer security incidents may be known, such as when a particular computer worm is known to exploit a specific operating system vulnerability, the number of computers containing this specific vulnerability and exploitable is often unknown. (The presence of the vulnerability may not be sufficient, host-based antivirus software may provide protection or additional user action may be required.) The causes of other types of incidents may never be determined and

could be attributable to several different possible security failings. In some cases, these incident types are only detected based on how the computers are being used following an attack (such as for relaying spam), and thus the cause of the initial exploit may remain unknown. However, occurrences of these incident types may still follow a pattern related to other factors behind attacks rather than only the distribution of vulnerabilities present in a population of computers.

One family of models comes from the field of software reliability. In the context of software reliability, software reliability growth models have been used to describe the software defect detection process. In general, two assumptions driving software reliability models are that software failures are caused by unpredictable events that do not have a corresponding remedy within the software and that the failures that occur are independent of each other. These two assumptions led to Goel and Okumoto's (G-O) model [23]. Many similar models have been proposed [24–26].

If the exploitation of vulnerable computers is viewed as being similar to the detection of faults in software, then software reliability growth models may have some advantages over other types of models for forecasting the level of future incident occurrence. In this view, attackers take the role of being testers of the software and identifying software faults present in a system. We apply some software reliability growth modeling concepts to computer security incidents to see if this view is useful.

Public health surveillance involves similar quality control and monitoring models and processes related to disease outbreaks in human populations. Time series analysis and models have been used as surrogates for the use of epidemic models for monitoring and control purposes. Similar to software reliability growth models,

time series models rely mostly on the time dependent structure present in a set of observations. While time series models are often encountered in economics and finance, they have also been described in [28] as a method to potentially identify developing outbreaks.

1.4.4 Selecting and targeting interventions

There are many interacting factors that affect when and where a computer security incident occurs. Understanding the underlying mechanisms of a computer worm or virus is only part of the puzzle. Characteristics of the population being attacked or affected are also important. This includes the distribution and prevalence of vulnerable operating systems and applications, differences in user behavior [29] in the type and frequency of use of the computer systems, and differences in local management of network and computing resources.

In the study of human diseases, the number of cases of a disease is often divided by some measure of population to obtain a rate of incidence. This incidence rate can then be used to compare the frequency of disease cases among different populations of varying size. Differences in incidence rates can sometimes be attributed to known differences of properties of the populations, such as genetics, environment, habits, etc.

For many diseases in humans, biological age is a known risk factor. It is common to calculate incidence rates for several age stratifications or groupings when making comparisons between two populations. This can help account for differences

in the total incidence rate of two populations with different age demographics. For example, if the total incidence rate for a type of cancer is higher for one population than another, the population with the higher rate of incidence may also consist of a higher percentage of older people (who may be more at risk for the type of cancer) and yet the age-specific incidence rates for the two populations may be similar.

In addition, some environmental factors can change over time or in cycles (such as seasonal weather patterns in some geographical climates). The social interaction among segments of the population can fluctuate as well (e.g. school related social interaction decreases during seasonal vacation breaks). These temporal factors can be observed in patterns of human disease such as the common flu. Note that for seasonal variations, time itself is not a risk factor but simply serves as a proxy indicator for other risk factors that track the seasonal changes.

There is a need for empirical examinations of risk factors relevant to computer security incidents using real-world data. This information can be useful for focusing intervention efforts where they may be most effective. Verifying or quantifying risk factors in actual environments can be challenging. Exploring the influence of some of these factors using field collected data can be an important step to build upon both for future simulations and for exploring other factors. Anbalagan and Vouk [30] present a security model based on vulnerability states and rates of disclosure, exploitation and correction tested with data.

1.5 Purpose

This research applies concepts and models from Epidemiology and Public Health frameworks to computer security. This includes the exploration and adaptation of compartmental SIR models often applied to modeling the spread and control of infectious diseases in human and animal populations. Additional methods for analyzing incident data for the purposes of monitoring and forecasting expected future incidence rates from the software reliability field are also covered. Comparing actual incidence rates to expected rates can indicate if additional intervention actions are needed and/or if existing intervention actions and preventative measures or policies are being effectively deployed. This research also includes examples of epidemiological type analysis for examining potential risk factors and risk markers for incident occurrence which may be useful for intervention targeting.

1.6 Significance

The significance of this research is in showing how existing models can be applied and adapted to cover computer security incidents. Some of these models can be used as tools to identify, evaluate and compare potential interventions. Modifications to some current models help to provide a more representative picture of underlying dynamics and may also indicate additional intervention opportunities for some incident types. This is useful to organizations deciding how to allocate resources towards intervention efforts. Other models can be used to monitor and

gauge the impact of enacted intervention efforts or policies. In turn, this can provide organizations with feedback regarding their intervention actions.

1.7 Scope and Limitations

This research is scoped to study computer security incidents that have occurred within one organization during a specified time period. The modeling of the incidents includes aspects of the targeted computers and makes some limited assumptions regarding user characteristics for some incident types. Aspects regarding attacker incentives or motivations are not included for these models. The application of models covers a range of incident types, as well as a range of potential targets, but they are not applicable to modeling incidents that result from a directed or targeted attack by a single or group of attackers.

Chapter 2: Literature Review

2.1 Overview

This chapter provides an overview of some relevant definitions and aspects of current research pertaining to computer security incidents. These aspects are important for modeling the occurrence of computer security incidents. One family of models explored is compartmental (Susceptible-Infected-Recovered [SIR] and variants) models based on modeling the spread of infectious diseases in human and animal populations. Some general types of interventions applicable to the control and containment of infectious diseases in human and animal populations are described along with equivalent type interventions applicable to some types of computer security incidents. Surveillance type models (software reliability growth and time series) often used for quality control and process monitoring are also covered. These types of models may be applicable to a broader range of incident types which may not be easily modeled as an infectious process; thus the impact of interventions may be more difficult to predict, but important to measure. Some weaknesses or gaps in current research knowledge are also discussed.

2.2 Computer Security Incidents

Although not required, many computer and network security incidents involve the exploitation of software related errors, either in the form of exploitable vulnerabilities or improper use and configuration. Howard and Longstaff define a vulnerability as "a weakness in a system allowing unauthorized action." This can include design, implementation or configuration of the software. Arbaugh [31] describes a vulnerability life-cycle model as one in which vulnerabilities are discovered and disclosed, patches are created and distributed, and unpatched systems are at risk of being exploited until patched. This model is further explored by Frei [32] and includes examination of the risk of time windows where a vulnerability has been discovered but remains undisclosed, and a patch or fix has not yet been made available.

While vulnerabilities in software may contribute to the prevalence of many types of computer or network security incidents, in some cases human users of these systems are exploited by an attacker to gain unauthorized access or to initiate an unauthorized action. Downs [33] describes some cognitive models to examine what contributes to a successful phishing attack. Jagatic et al [34] also identify some tactics used to trick users via email and mention some potential defenses. Brown et al [35] warn that publicly available information regarding both social connections and personal information can be used to craft more personalized and potentially more successful email attacks against a large percentage of users. Fette et al [36] propose a machine learning approach for filtering out phishing type attacks, while Herley [37] suggests that users should not bother to determine if every email message

is legitimate or not due to the time costs outweighing the potential benefits.

Weaver et al [38] list several ways incidents and attacks may be triggered. Some attack types and incidents may be independently initiated, while others may involve a form of self-replication where compromised systems autonomously initiate attacks against other users or systems. The method of propagation can be important in determining the best types of models and interventions applied towards an incident type.

2.3 Infectious Disease Modeling

Modeling the propagation of network worms as analogous to an infectious disease process has been applied to events such as the Code Red worm, as in the work of Zou et al [39] and Chen et al [40]. Tailored defense strategies for this class of malware have also been explored. Zou et al [22] propose the short-term quarantining of suspected infected hosts to slow the spread of infected hosts, while Sidiroglou and Keromytis [41] describe automated patch development based on deployed sensors and heuristics identifying exploited vulnerabilities.

While several previous works have focused on modeling incident types that spread over traditional wired computer networks, additional research has examined some of the ways propagation over wireless connections affects the network topology and therefore the spread of malware. Yoneki et al [42] examine the spread of an infection in a time-dependent dynamic human network. Nodes would be in contact with other nodes for changing periods of time throughout the day based on changing

physical proximity. This differs from the standard assumption of nodes having the potential for constant contact with each other in a wired environment. Milliken, et al [43] model the spread of a virus that infects wireless access points in its vicinity through firmware modification. One reported finding is that in urban environments, the density of wireless access points impacted the spread of the virus more than susceptibility of a particular access point. They also describe two algorithms for detecting such attacks in a laboratory environment.

2.3.1 Standard Susceptible-Infected-Recovered (SIR) model

Most of the classic compartmental infectious disease models, such as the Susceptible-Infected-Susceptible (SIS) model and the Susceptible-Infected-Recovered (SIR) model, are based on the assumption of a fully connected network topology (each infected host has the ability to contact and infect any other susceptible host in the population). This is different from some ecological models (such as the forest-fire model, which limits contact to adjacent neighbors on a two-dimensional grid). The impact and effect of different network topologies has also been explored in the context of the spread of infectious diseases by Pastor-Satorras and Vespignani [44], Keeling [45], and Danon et al [46], as has computer worm propagation by Kim et al [21].

A property of infectious disease models that differentiates them from chronic disease and other models is that the chances of a host becoming infected (transitioning from the "Susceptible" compartment to the "Infected" compartment in an SIS or SIR

compartmental model) depends on the number or proportion of other infected hosts in the population. In most cases, these models assume that other than the initial infection(s), any additional infected hosts result from contact with other infected hosts within the population and are referred to as secondary infections.

However, a population may not be completely isolated from other sources of infection, and additional hosts may become infected by exposure to external sources (referred to as reservoir) and not by exposure to other infected hosts within the population. An example of this is when a disease can spread from animals to humans and also from humans to humans. Ongoing contact with animals can provide a source of additional primary infections to the human population, as is explored by Singh et al [47]. Differentiating the proportion of primary and secondary infections based on the distribution in time of identified infected cases may not be feasible, as is demonstrated by Singh et al [47]. In these cases, mitigation strategies need to address both the primary causes (contact between susceptible hosts and an infected reservoir) and the secondary causes (contact between susceptible and infected hosts within the population).

2.3.2 Model for email propagation

While some of the assumptions included in the standard epidemic models may be applicable to malware and security incidents that spread directly from device to device, these assumptions may not be as applicable to computer viruses that spread through email. The timing delays between when an infected message is received and

when it is accessed or read may alter the overall pattern of spreading, as is illustrated by Vazquez et al [48].

Other factors related to how users interact with email may also influence the spreading behavior of an email virus. A common strategy used by email viruses is to send out infected messages to any stored contacts in an infected user's address book. The number and interconnectedness of address book contacts largely creates the topology of the network such a virus will use for spreading. This type of contacts based topology has also been explored in the context of worms that may spread by gathering targets from SSH "known_hosts" files, as demonstrated by Schechter, et al [49]. Also relevant to email virus propagation is the likelihood that a user will open an infected message on a vulnerable computer. Gao et al [50] describe a model that incorporates these factors and explores the impact of interventions which target nodes in the contacts network based on their level of connectedness.

2.3.3 Types of interventions for controlling infectious diseases

Options for controlling the spread of infectious diseases in humans and/or animals typically work by limiting the transmission between infectious and susceptible hosts. Interventions include: (a) vaccination; (b) quarantining; and (c) culling. Vaccination attempts to transition susceptible hosts directly to the recovered state without becoming infected. This reduces the number of susceptible hosts that can become infected. Quarantining isolates known or likely infected hosts from other susceptible hosts. This reduces the number (or proportion) of infected hosts in

contact with susceptible hosts, which also reduces the likelihood of transmission to a susceptible host. Culling is a strategy sometimes used with rapidly spreading animal or plant diseases. It involves reducing the overall size of a population by removing (or killing) both infected and susceptible hosts, but mostly in limited or targeted areas. Culling reduces both the number of infected, as well as susceptible, hosts. Keeling and Rohani [20] include a more detailed description of these interventions and their applications to human and animal infectious diseases.

There are similar or analogous options for controlling the spread of network worms in a computer population. Vulnerable systems can be patched or updated so they are no longer susceptible (analogous to being vaccinated), and network connectivity can be disabled for systems identified as being infected (analogous to being quarantined). Perhaps not as drastic as culling, an equivalent measure could involve preemptively isolating critical hosts or subnets until other corrective actions can be taken (such as either patching systems or migrating functionality to non-vulnerable systems). This also reduces the size of the susceptible population.

Other options for controlling the spread of network worms in a computer population include the use of anti-virus software to detect and block exploit actions on vulnerable hosts as well as the use of a network based IPS to detect and block potential exploit traffic on its way to systems (which may be susceptible, already infected, or already recovered).

2.4 Surveillance–quality control and monitoring

Public health surveillance is often defined as "the ongoing systematic collection, analysis, and interpretation of data, closely integrated with the timely dissemination of these data to those responsible for preventing and controlling disease and injury" as stated in Klaucke et al [51]. A World Bank publication (Nsubuga et al [52]) states:

> *Because surveillance can directly measure what is going on in the population, it is useful both for measuring the need for interventions and for directly measuring the effects of interventions. The purpose of surveillance is to empower decision makers to lead and manage more effectively by providing timely, useful evidence.*

According to a Center for Disease Control (CDC) publication [53], originally public health surveillance involved the monitoring and tracking of communicable and infectious diseases; however, it is not limited to this and can also include the monitoring and tracking of injuries, birth defects, chronic diseases, and health behaviors (such as diet, amount of sleep, and frequency of exercise).

Table 2.1 shows a list of purposes for monitoring health events from a CDC publication [53] in the left column. The right column shows how each can be easily adapted to apply to monitoring computer security incidents. Just as public health surveillance is applied to help monitor, control, and understand the spread and growth of diseases in human populations, a similar approach can be applied by organizations to help monitor, control, and understand the occurrence and growth

Table 2.1: Purposes for monitoring health events and computer security incidents.

Purposes for monitoring health events:	**Purposes for monitoring computer security incidents:**
• To detect sudden changes in disease occurrence and distribution	• To detect sudden changes in incident occurrence and distribution
• To follow long-term trends and patterns of disease	• To follow long-term trends and patterns of incident occurrence
• To identify changes in agents (such as the flu virus) and host factors (such as smoking, alcohol use, and seat-belt use).	• To identify changes in agents (such computer viruses) and host factors (such as use of host-based firewalls, host-based AV, peer-to-peer usage).
• To detect changes in health care practices	• To detect changes in IT practices

of computer security incidents within their organization.

Based on the nature of an incident, epidemic models may not be suitable for some computer security incident types. Applying other types of models developed based on the analysis of security incident data can be used by organizations to assist with the efficient allocation of resources and to provide feedback regarding the impact of prevention and control measures or changes to security policies. One way this can be accomplished is by comparing model forecasts to observed activity. If the observed number of incidents begins to deviate as an increasing trend away from the forecast values, an external factor may have changed (such as the development of a new variant of an attack), and an organization should reevaluate its control measures and make necessary adjustments.

A similar strategy can be used for evaluating the impact of changes to the internal security environment (such as policy changes, IDS/IPS deployment, system updates, etc.). Model forecasts can be compared to actual observations before and after implementing a new control measure. The effectiveness of the change to the

security environment can be gauged if the observations trend away from the forecast values after the new control measure has been implemented. Although these types of models do not provide as much insight into the underlying dynamics leading to incident occurrence as epidemic models, they can be more easily applied to a wider range of incident types and interventions and require less information to develop.

2.4.1 Software reliability growth models

In the context of computer security, two phenomena have been commonly modeled using software reliability growth models: the trend of exploitations by Browne et al [54] and Rescorla [55] and the vulnerability discovery process by Alhazmi and Malaiya [56,57]. Regarding exploitations, Browne et al [54] used simple mathematical models to fit computer security exploits. They used exploit data from the reports repository of the Computer Emergency Response Team (CERT). Some delays may have occurred between the time the exploit was initiated and the time the exploit was reported to CERT. Exploits were also reported by a wide range of users; thus there is limited control over the data collection method.

Regarding modeling the security vulnerability discovery process, Anderson [58] proposed an equation to estimate the probability of system failure as a result of the vulnerabilities. Alhazmi and Malaiya [56] examined the vulnerability discovery process in operating systems and treated vulnerabilities as equivalent to software defects. A few models were used to fit the vulnerability detection data, including the software reliability growth Goel-Okumoto (G-O) model. The authors indicated

that if a model can accurately fit the observed data, then it could predict the future vulnerability detection trend. Alhazmi and Malaiya [57] also examined the vulnerabilities in the Apache and IIS HTTP servers. The Linear Vulnerability Discovery model (LVD) and the Alhazmi-Malaiya Logistic model (AML) were used to fit the vulnerability detection data and predict the future vulnerability discovery.

2.4.2 Time series models

Although epidemic models have been used in the past to model the spread of incidents, there may be other factors not included in such models (such as the academic calendar in a university setting) that can cause fluctuations in populations and affect incident occurrence. Different interconnection topologies can result in varying epidemic patterns even when other factors are the same, as shown by Watts et al [59], and new strains (or variants) of viruses can emerge, thereby rendering existing vaccinations less effective. Models that attempt to incorporate or account for the influence of these factors become increasingly computationally complex. Also, when it comes to modeling computer security incidents, some incident types may not be aptly described as or result from a contagious process.

Another approach is to use time series analysis and modeling as a surrogate for epidemic models for control purposes or to include the influence of factors not easily incorporated into or applicable to epidemic models. Time series models are not explanatory models and do not rely on knowing the underlying dynamics producing the observed behavior. Instead, time series models attempt to make use of the

time dependent structure present in a set of observations. This method is often encountered in the fields of economics and finance, and it has been described by Allard [28] as a general technique for the early identification of outbreaks of infectious diseases with applications for intervention strategies. The use of time series modeling is also illustrated by Trottier et al [60] when applied to childhood infectious disease (pertussis, mumps, measles and rubella) data from Canada to gauge the impact of mass vaccination. Lai [61] describes monitoring the SARS epidemic in China using time series models, and Han and Leong [62] describe similar methods to evaluate intervention measures taken in Singapore in response to SARS. Helfenstein [27] suggests that classification of infectious diseases may be possible based on their time series structure.

2.5 Targeting Interventions and Identifying Risk Factors

Often in human and animal populations, the risks of a particular individual being affected by a disease (infectious or not) are not uniform throughout the population. Factors such as age (with which susceptibility may vary) or geography (which may affect exposure to contributing elements) can influence the risk of being infected or becoming sick. In the case of an infectious disease, interventions may be targeted to include groups more at risk or more likely to spread the disease to other groups. This is an effort to minimize the spread of a disease while also efficiently using available resources (such as available doses of a vaccine as well as deployment of qualified administrators of the vaccine). In the case of a non-infectious disease,

identifying risk factors can help direct intervention efforts towards groups with higher risks even if underlying root causes are not well understood.

Michael et al [63] demonstrate the use of mathematical models to compare the use of different drug combinations for controlling lymphatic filariasis (a parasitic tropical disease carried by mosquitoes). Models were used to compare the potential effectiveness of different control strategies, and it was determined that an optimal strategy would need to include aspects for controlling the parasite itself as well as controlling the transmission vector (mosquito population). Models were used to support decisions regarding different intervention strategies to reduce the number of disease cases within a desired time frame.

Hay and Ward [64] provide another example of applying disease models in the context of decision support. They compare costs and effectiveness for applying pertussis vaccinations to different age groups in the United States. Their cost-benefit analysis concludes that there is an overall benefit to immunizing adolescents (ages 10-19) with a pertussis booster vaccine. The analysis includes costs related to vaccination, as well as costs resulting from pertussis and pertussis-related complications.

Hoggart et al [65] have developed a risk model for lung cancer. One of the proposed applications of such a model is to identify high risk individuals who could be encouraged to take steps to lower their risks. High risk individuals could also be identified to undergo increased monitoring by clinicians or be selected for participation in clinical trials or treatment programs. The model includes lifetime exposure to cigarette smoke, age effects (including ages of initiation and cessation of smoking) and duration of smoking. Similarly, identifying risk factors and developing a risk

model for computer security incidents could provide guidance for the targeting of intervention actions.

2.6 Existing Limitations

While the current research provides a good description of some of the problems related to modeling and preventing computer security incidents, there are also some weaknesses or gaps to be addressed. One potential intervention of dynamic quarantine described by Zou et al [22] assumes such an intervention could be applied or implemented to an entire population. If so, this type of intervention could slow the propagation of an Internet worm. However, implementing such an intervention on such a large-scale is unlikely, as it would involve the coordinated action of many separate entities and organizations. It is unknown at what scale such an intervention would need to be applied in order to see effects beyond the subpopulation to which it has been applied.

The intervention explored by Gao et al [50] in response to a virus that propagates using email examines the impact of targeting highly connected nodes (individuals with large email address books). However, this assumes more complete knowledge of interpersonal email network connection topology information than is typically available. Also, their modeling efforts often relied on the assumption of a standard normal distribution for human related factors (such as email checking frequency and likelihood of opening infected messages). It is unknown if the assumption of other distributions for some of these factors would lead to different observations or

conclusions.

In several cases where epidemic models were used to explore observed behavior and novel interventions, the utility or impact of existing or commonly available interventions were not included for comparison. This leaves open whether or not some of the proposed interventions are feasible or practical from a cost perspective, or if they provide additional improvement to some more commonly available strategies. Cost may not be an easy aspect to definitively measure, but outcomes of strategies can be compared to provide to some guidance.

In terms of targeting intervention efforts, little research seems to be available regarding population type risk factors for incident occurrence. While each incident type likely disproportionately affects a certain subpopulation, information for generally identifying higher risk individuals is lacking. It is possible that due the changing and dynamic nature of attacks and defenses, stable risk factors may not be easy to identify or may not provide much utility for predicting future risk.

Chapter 3: Infectious Disease Models

3.1 Overview

In this chapter, we describe some infectious disease models and illustrate some aspects and modifications relevant to modeling some types of computer and network security incidents in an interconnected population of computers or devices. Infectious diseases are typically modeled using both deterministic and stochastic models. Deterministic models are more appropriate when modeling large populations and the law of mass action applies. Stochastic models may be more applicable for modeling smaller populations and for capturing variability due to randomness of interactions or other factors. Stochastic models can also be more flexible for incorporating features that may not adhere to assumptions made by deterministic models (such as assuming a fully connected network topology). We review a standard deterministic model and describe modifications we make to a particular stochastic model in order to apply it to modeling the spread of some types of computer security incidents and to include features linked to intervention actions.

3.2 Deterministic SIR Models

The Kermack-McKendrick model [66] is a deterministic model that allows hosts to be designated as part of three compartments or states: Susceptible (S), Infected (I), or Removed/Recovered (R). The model specifies the transition of hosts between states (from $S \to I$ and from $I \to R$) where the recovered state is terminal (hosts do not transition to another state after being recovered), and some hosts may always be in the susceptible state. It is often referred to as the classic SIR model and specified as:

$$\begin{aligned} \frac{dS(t)}{dt} &= -\beta \frac{S(t)I(t)}{N} \\ \frac{dI(t)}{dt} &= \beta \frac{S(t)I(t)}{N} - \gamma I(t) \\ \frac{dR(t)}{dt} &= \gamma I(t) \\ N &= S(t) + I(t) + R(t) \end{aligned} \tag{3.1}$$

where β is the infection rate ($S \to I$) and γ is the recovery rate ($I \to R$) and N is the total size of the population being considered. The infection and recovery rate parameters are assumed to remain relatively constant or fixed in time for the interval being modeled.

An extension of the classic SIR model allows for a direct transition from Susceptible to Recovered ($S \to R$) without becoming infected. This accounts for the

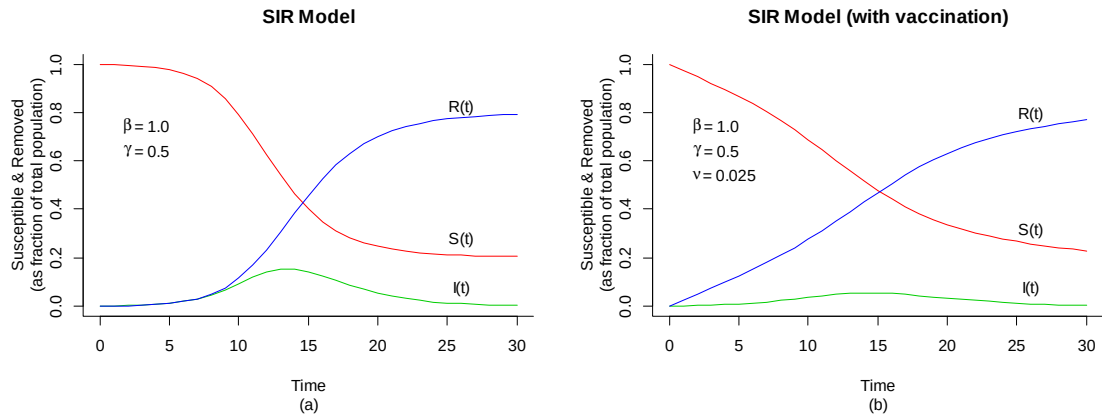

Figure 3.1: Deterministic examples of (a) an SIR model and (b) an SIR model with vaccination.

possibility of being vaccinated. This model is specified as:

$$\begin{aligned} \frac{dS(t)}{dt} &= -\beta\frac{S(t)I(t)}{N} - \nu S(t) \\ \frac{dI(t)}{dt} &= \beta\frac{S(t)I(t)}{N} - \gamma I(t) \\ \frac{dR(t)}{dt} &= \gamma I(t) + \nu S(t) \\ N &= S(t) + I(t) + R(t) \end{aligned} \tag{3.2}$$

where β is the infection rate ($S \to I$) and γ is the recovery rate ($I \to R$) and N is the total size of the population being considered, and with ν as the vaccination rate ($S \to R$). Figure 3.1 shows examples of (a) an SIR model and (b) an SIR model with vaccination.

3.3 Stochastic SIR Models

While a deterministic SIR model summarizes the overall behavior of the population, a stochastic SIR model allows for modeling the state of each member

of the population and allows for variations in the outcomes of repetitions using the same parameter values. This can be useful for constructing a distribution of outcomes and evaluating a "worst-case" scenario rather than limiting forecasts to average expected behavior. Stochastic models can also be more flexible for exploring different connection (or mixing) topologies. Since the state of individuals is being modeled, it is also possible to specify non-uniform connection topologies without knowing beforehand the effects on the aggregate behavior [44], [45], [46].

A stochastic algorithm for calculating a compartmental SIS model [where infected nodes can return a Susceptible (S) state rather than to a terminal Recovered (R) state] with network topology specified in the form of an adjacency matrix is presented in [67]. We adapt this algorithm to simulate an SIR model, the assumption being once an infected host has been identified and removed, it will only be replaced if it is no longer susceptible to the original source of infection. First we review the described SIS model and the presented algorithm, and then we explain the changes made to simulate an SIR model using stochastic modeling code we created (implemented in the R [68] programming language).

The standard SIS model involves two states: Susceptible (S) or Infected (I) and there can be transitions from $S \to I$ as well as from $I \to S$. The deterministic model can be specified similar to the classic SIR model, except there is no terminal Recovered (R) state:

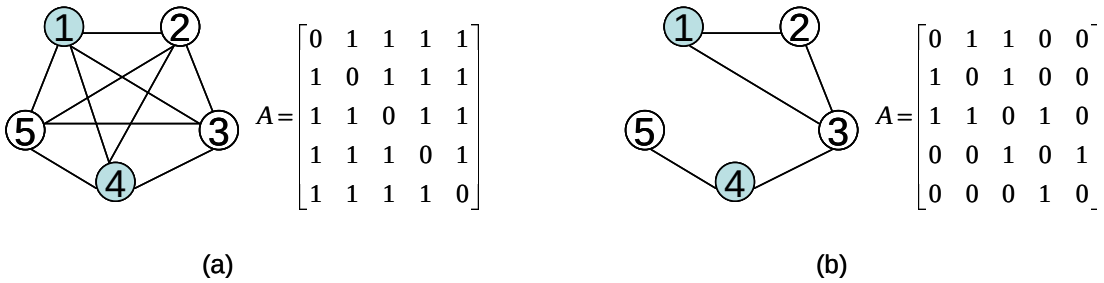

Figure 3.2: Examples of adjacency matrices for (a) fully connected graph and (b) partially connected graph.

$$\begin{aligned} \frac{dS(t)}{dt} &= \gamma I(t) - \beta \frac{S(t)I(t)}{N} \\ \frac{dI(t)}{dt} &= \beta \frac{S(t)I(t)}{N} - \gamma I(t) \\ N &= S(t) + I(t) \end{aligned} \tag{3.3}$$

where β is the infection rate ($S \to I$) and γ is the recovery rate ($I \to S$) and N is the total size of the population being considered.

Instead of allowing for complete mixing of the population as is done in the deterministic model, the presented stochastic algorithm uses a defined adjacency matrix A to represent how members or nodes of the network are connected to each other and can interact. Figure 3.2 shows simple examples of two networks and their corresponding adjacency matrices. (Note that each adjacency matrix is symmetrical across its diagonal; this is because we are assuming bidirectional links. However, this is not required and there may be instances where breaking such symmetry is applicable.)

If v is a state vector at time t where v_i is 1 if node i is infected and 0 if node

i is not infected (and therefore susceptible), then the product components of Av contain the number of infected neighbors of each node at time t. For example, if nodes *1* and *4* from Figure 3.2(a) are infected, then:

$$Av = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 1 \\ 2 \end{bmatrix} \tag{3.4}$$

indicating nodes *2*, *3* and *5* each have two infected neighbors, while nodes *1* and *4* each have one infected neighbor (although they are infected themselves, they do not count as their own neighbor). However, if nodes *1* and *4* from Figure 3.2(b) are infected, then:

$$Av = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 2 \\ 0 \\ 1 \end{bmatrix} \tag{3.5}$$

so only node *3* has two infected neighbors, while nodes *2* and *5* each have one infected neighbor.

An infection (the transition of a node from $S \rightarrow I$) is modeled as a Poisson process with rate β, where the probability a susceptible node becomes infected in the time interval $(t + \Delta t)$ is approximated by $(\beta * K/m) * \Delta t$, where K is the number

of infected neighbors for the node and m is the average degree of the network (or average number of neighbors for each node). At each time step, the algorithm does the following:

- For each susceptible node, generate a uniform random number "RAND" in the interval [0,1];
- If: RAND $< (\beta * K/m) * \Delta t$, then change state of susceptible node to infected;
- Repeat for all susceptible nodes.

For the stochastic SIR model with vaccination, our algorithm does the following:

- Add the terminal Recovered (R) state;
- Keep the same susceptible-to-infected ($S \rightarrow I$) transition used in the stochastic SIS model;
- Change the SIS infected-to-susceptible ($I \rightarrow S$) transition to be the SIR infected-to-recovered ($I \rightarrow R$) transition;
- Add a vaccination or susceptible-to-recovered transition ($S \rightarrow R$);
- At each time step, state transitions are checked in the following order for all nodes: $S \rightarrow I$, then $I \rightarrow R$, then $S \rightarrow R$. (Only one change of state is permitted for nodes at each time step.)

Figure 3.3 shows examples of (a) a stochastic SIR model and (b) a stochastic SIR model with vaccination. For the SIR model with vaccination, Figure 3.3(c) shows

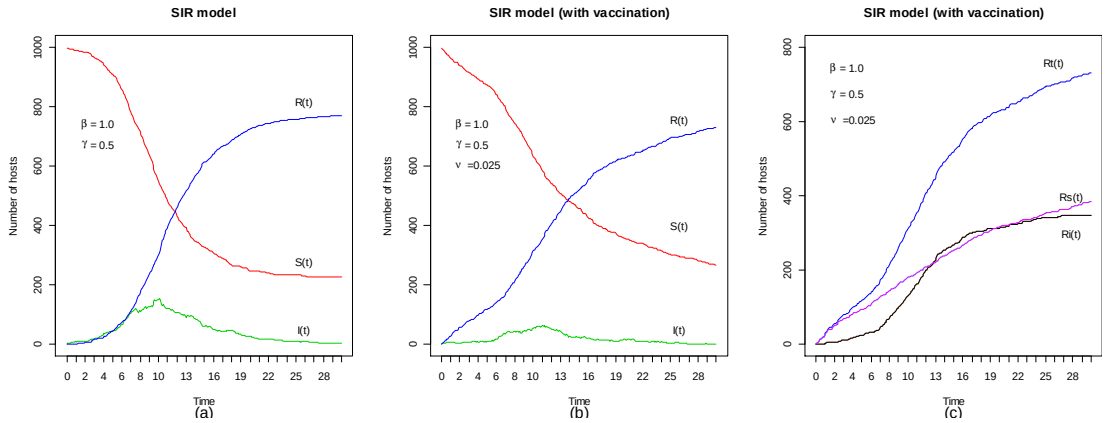

Figure 3.3: Stochastic examples (with $\Delta t = 0.1$) of (a) SIR model and (b) SIR model with vaccination, (c) breakdown recovered hosts as originating from infected hosts $[R_i(t)]$ or vaccinated hosts $[R_s(t)]$.

identifying the number of recovered nodes that transitioned from originally susceptible $[R_s(t)]$ or originally infected $[R_i(t)]$ states. Although a vaccination rate (ν) may be lower than the recovery rate (γ), since the number of susceptible nodes is often larger than the number of infected nodes, there are often more recovered nodes originating from the susceptible state than the infected state. Our stochastic models have been implemented using the R [68] programming language. An example of the R code used for the SIR models is provided in Appendix B.

3.4 Isolating a Smaller Subset from a Larger Population

Often when epidemic models are used to examine the propagation and mitigation of network worms, there is an assumption that intervention actions can be implemented on a scale affecting the whole population being modeled. However, in many cases, individual organizations or businesses, while inter-connected with a larger population, can only take steps to control malware that is spreading on

the subset of the population that is under their direct authority or control. For example, a university may be allocated a Class B IPv4 network range (≈65,000 IPv4 addresses) and can take steps to remediate and control the spread of network worms in this range of IP addresses, but it would be difficult to apply such measures to the rest (≈4 billion) of IPv4 connected devices elsewhere in the world. Even within an organization, it is possible for IT resources to be managed and controlled separately by different departments or divisions. So while a network worm may be spreading globally, efforts to contain it are likely to be uncoordinated and focused on smaller subsets of a larger population. For incidents observed within the smaller subset of the population, it may not be easy to determine how many resulted from connections with externally infected hosts and how many resulted as secondary infections from other internally infected hosts within the subset. This could be important information to have when assessing mitigation options and strategies.

In this following scenario, we explore some of the properties that might be observed from the perspective of a smaller population subset within a larger interconnected population. To accomplish this, we make some additional changes to our adjacency matrix and to our algorithm for the stochastic SIR model described earlier. Instead of using a single fully connected adjacency matrix A, we split it into two component adjacency matrices. One component adjacency matrix A_{ext} allows for full interconnections between nodes not in the smaller population subset, as well as between nodes in the smaller population subset and the rest of the larger population. However, it does not include any interconnections between nodes only contained within the smaller subset. The other component adjacency matrix A_{int} only allows

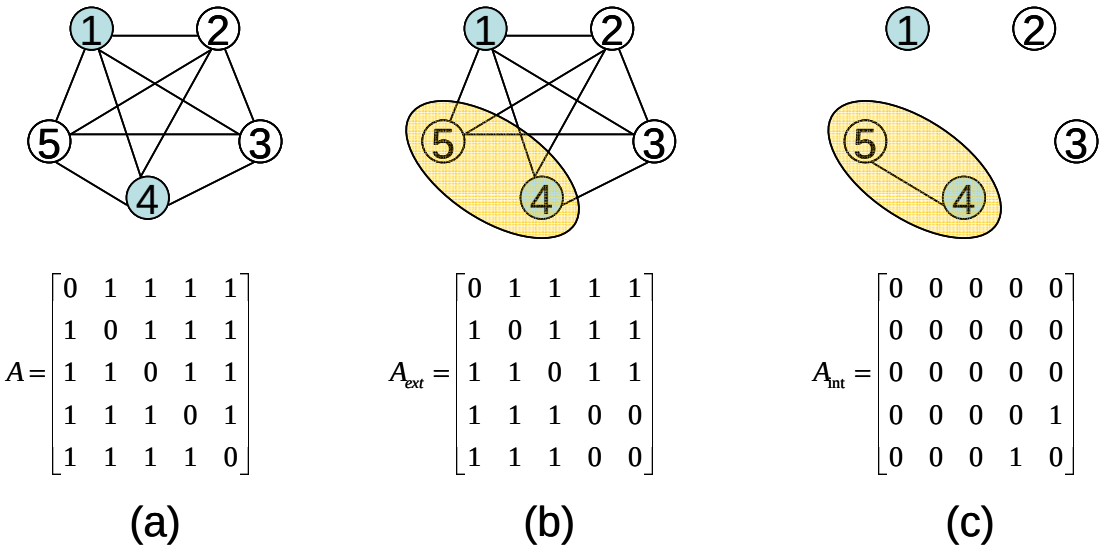

Figure 3.4: Example of decomposing (a) fully connected graph and adjacency matrix $[A]$ into parts (b) representing connections to/from outside $[A_{ext}]$ a defined subset (nodes *4* and *5* in example) and (c) representing only internal connection $[A_{int}]$ of the subset where $A = A_{ext} + A_{int}$.

for full interconnections between nodes within the smaller population subset and does not allow for any connections between nodes in the subset and the rest of larger population. Figure 3.4 shows (a) a fully connected network and its adjacency matrix A along with the graphs and adjacency matrices for its components A_{ext} and A_{int} based on an example population subset (defined as containing nodes *4* and *5* as shown in Figures 3.4(b) and 3.4(c)).

With two adjacency matrices, we can now calculate K as having two components as well where $K = K_{ext} + K_{int}$ and values for K_{ext} and K_{int} are obtained from $A_{ext}v$ and $A_{int}v$, respectively. For a given node, K_{int} is the number of infected neighbors that belong to the smaller subset of the population, while K_{ext} is the number of infected neighbors that are not part of the smaller subset of the population. Since K has now been split into two components, our algorithm step for checking transitions

from $S \rightarrow I$ must also be changed to reflect this and it becomes:

- If: RAND $< (\beta^*(K_{ext} + K_{int})/m)^*\Delta t$, then change state of susceptible node to infected.

The result of this transition check can be compared to the result of an additional check made at the same time (such that it uses the same "RAND" value):

- If: RAND $< (\beta^*(K_{ext})/m)^*\Delta t$, then change state of susceptible node to infected.

If both of the above transition checks have the same result, then any infected neighbors that are part of the smaller population subset had no impact on the result. However, if there is a difference between these transition checks, then contact with an infected node in the smaller population subset is responsible for the infection and it can be considered a secondary infection. Making this distinction allows us to determine which of the infections within the subset were caused by sources outside the subset and which infections were secondary infections caused by sources within the subset.

Figure 3.5 shows (a) an example of a stochastic SIR model behavior in a subset (10%) of the total population, (b) the total number of infected hosts in the subset population as well as how many infected hosts were due to contact with only external or internal infected hosts (c) a comparison of number of unique infected hosts in the subpopulation with causes internal and external to the subpopulation. Table 3.1 summarizes some comparisons for stochastic models using different parameter values and population sizes. For a fully connected population, the ratio of internally caused

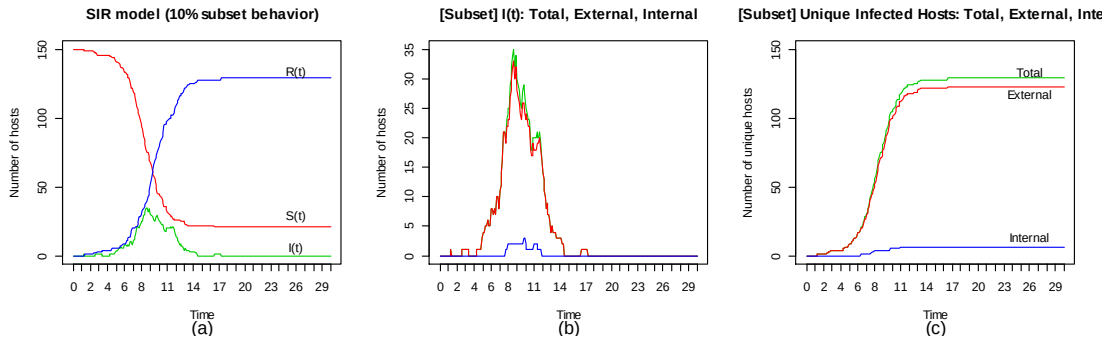

Figure 3.5: Contribution of subpopulation to overall number of infected hosts.

Table 3.1: Comparisons using different parameters and population sizes.

Total Population	Subpop	β	γ	Total Inf.	Subpop Inf.	Subpop External Inf.	Subpop Internal Inf.	(Infected) Subpop/ Population	(Infected) Subpop Int/Subpop
1,000	100	1.8	0.75	857.4	86.5	78.3	8.2	0.101	0.095
10,000	100	1.8	0.75	8537	85.2	84.2	1.0	0.010	0.012
10,000	1,000	1.8	0.75	8528	854.6	770.9	83.7	0.100	0.098
1,000	100	1.2	0.90	341.8	33.4	30.1	3.3	0.098	0.099
10,000	100	1.2	0.90	2070	20.7	20.5	0.17	0.010	0.008
10,000	1,000	1.2	0.90	1643	162.2	146.6	15.4	0.099	0.095

infections within a subset of the population to the size of the population subset typically reflects the same ratio as the size of the population subset to the full population. However, this ratio and its variability could change for other than fully connected network topologies.

Splitting the adjacency matrix into components (A_{ext} and A_{int}) as described above allows for another observation to be made. In some cases, the value of K could be viewed as analogous to or an indicator of a network attack where an infected computer attempted to contact and compromise another computer or device. In these cases, the sum of the values of K_{ext} across all nodes within the subset represents the number of attacks or attack activity as seen from the perspective of the smaller population subset. This is similar to activity that might be observed by an IDS/IPS that watches network traffic on the border between an organization's network and

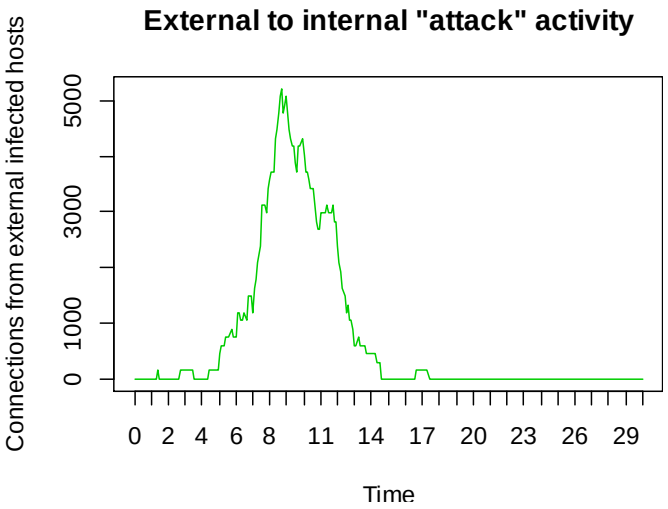

Figure 3.6: Number of infection attempts from external to internal hosts.

the rest of the Internet. Figure 3.6 shows an example of K_{ext} over time. Again, with the simple example using fully connected populations, the results may only be illustrative. However, other examples with different connection topologies may show different patterns of attack activity and it may be possible to use these observations to help infer properties of a network worm and the impact of external intervention efforts.

Before examining impact of different interventions, there is one more property to explore. Assuming that cases of initial infections originate outside of the smaller population subset, how long will it take for the first case(s) of infection to occur within a particular subset? The answer is likely to be dependent on the relevant rate parameters for a particular worm or virus as well as the ratio of population sizes (between the size of subset and the size of the overall population) and we can illustrate this using our models. Time can be an important factor when evaluating intervention options. IDS/IPS and anti-virus vendors usually need some time to develop effective

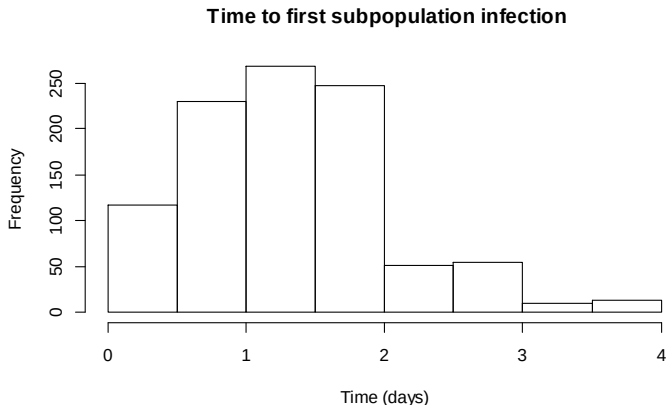

Figure 3.7: Time to first infection in sub-population. [Population = 1,000; sub-population = 100; $\beta = 1.8$; $\gamma = 0.75$].

signatures for identifying exploit activity while efforts to patch vulnerable systems may also require time for preparation and testing before implementation. In some cases, vendors of IDS/IPS products may release signatures earlier for premium or paying customers, while non-paying customers could be relying on community released signatures, which may take longer to become available. Being able to model the expected results of such intervention delays and expected activity may help determine if paying for early access to signatures is cost-effective for a particular organization.

Figure 3.7 shows the distribution of times when an infection first occurred within the subset of the smaller population for several runs of the stochastic model with fixed parameters. Table 3.2 summarizes the expected time to first infection (TTFI) in the smaller population subset for models using different parameters and population sizes. The results show an increase in time to first infection for smaller population subset sizes and a decrease in time to first infection with higher ratios of β/γ (which indicates an infection may spread faster than infections are removed).

Table 3.2: Time to first infection (TTFI) in internal population.

Population	Subpop.	β	γ	$R_0(\beta/\gamma)$	TTFI
1,000	100	1.8	0.75	2.4	1.6 ± 0.8
10,000	100	1.8	0.75	2.4	3.6 ± 1.9
10,000	1,000	1.8	0.75	2.4	1.6 ± 0.9
1,000	100	1.2	0.8	1.5	2.2 ± 1.2
10,000	100	1.2	0.8	1.5	5.7 ± 4.4
10,000	1,000	1.2	0.8	1.5	2.1 ± 1.2

Next we will examine the impact of some of the different interventions available. The two types of interventions we will be modeling are patching of vulnerable systems (modeled as vaccination) and IPS blocking at the border between the subset population and the rest of the population. The blocking activity at the border is modeled by specifying some permeability factor in the range $[0, 1]$ where *0* means all worm attack traffic is blocked, *1* means no worm attack traffic is blocked and values in between mean some worm attack traffic is blocked between the population subset and nodes outside of the subset. Effectively, this parameter scales or reduces the value of β for externally infected neighbors of the internal (subset) population, but β remains unchanged for infected neighbors of external nodes to each other or for internally infected nodes with other internal nodes.

When gauging the impact outcomes of different intervention strategies, we consider two aspects. The first is the number of infections and the second is the total time of infection across all nodes in the population subset. The second aspect reflects the area under the infection curve. The reason for this inclusion is that,

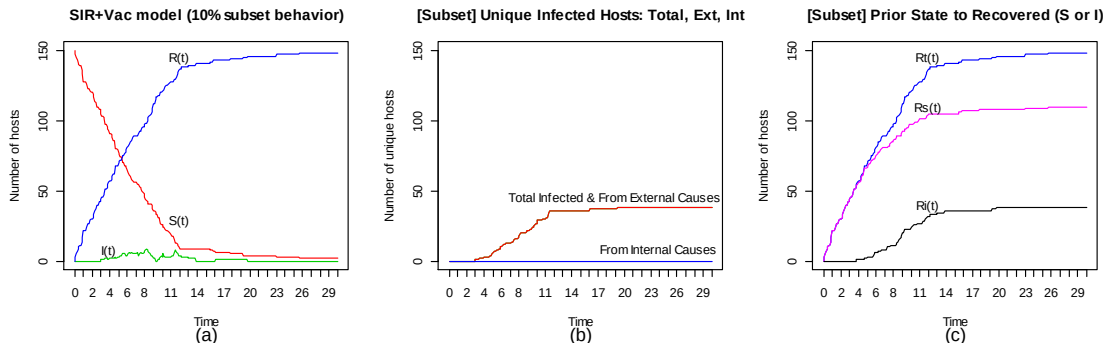

Figure 3.8: Example of vaccination only within a subpopulation.

while there may be somewhat uniform costs for recovering or remediating infected computers, there may also be additional risks the longer computers remain infected. Compromised computers may become joined to a botnet and engage in additional attacks against both internal and external information resources.

Figure 3.8 shows an example of patching vulnerable systems (of the population subset only) at a certain rate while Figure 3.9 shows an example of blocking infectious traffic between internal and external nodes of the population subset. Table 3.3 summarizes some of the outcomes of different patching or blocking rates. The results shown in Table 3.3 illustrate that intervention actions applied only to the smaller population subset can reduce the number of infections within the subset. Depending on the relative size of the subset, these actions may also reduce the number of infections outside the subset. Combining interventions also shows a positive interaction effect on reducing the number of infections in the subset.

We also make and explore a modification to the typically modeled vaccination or patching transition. Just as the infection rate depends on number of infected systems, we explore changing the vaccination rate so that it depends on the number

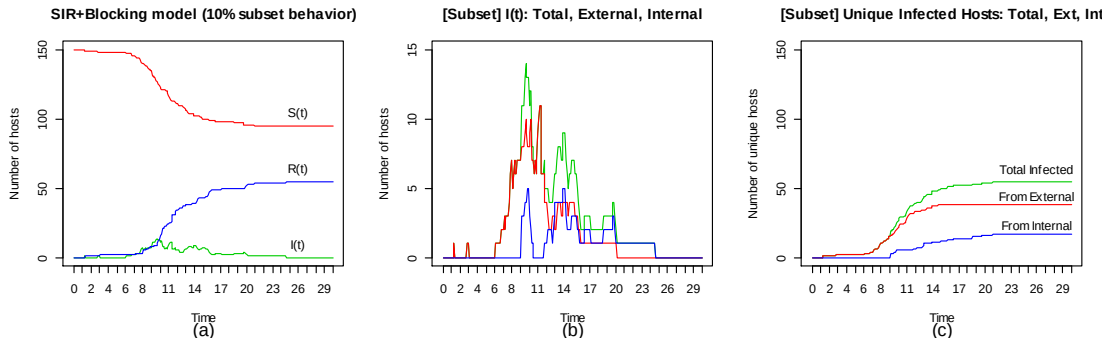

Figure 3.9: Example of blocking some externally infection connections.

Table 3.3: Summary of different outcomes based on vaccinations or blocking values.

Population	Subpop	β	γ	Vac. Rate	Blocking	Total Inf.	Sub Inf.	Sub Inf. Time
1,000	100	1.8	0.75	0.0	1.0	853	85.6	104±16.2
1,000	100	1.8	0.75	0.1	1.0	803	46.9	56.6±12.5
1,000	100	1.8	0.75	0.0	0.2	784	34.6	43.6±9.7
1,000	100	1.8	0.75	0.1	0.2	752	14.4	16.7±7.1
10,000	1,000	1.8	0.75	0.0	1.0	8,524	852	1,057±46
10,000	1,000	1.8	0.75	0.1	1.0	7,761	345	423±66
10,000	1,000	1.8	0.75	0.0	0.2	7,732	328	410±38
10,000	1,000	1.8	0.75	0.1	0.2	7,390	105	130±29

of susceptible systems. The rationale behind this is that the effort to patch vulnerable systems is not uniform and that some systems will be patched earlier and at a quicker rate which will then decline as there are fewer susceptible systems to patch and the ones remaining may require more effort to patch and will therefore be patched at a slower rate. This is implemented by changing the susceptible-to-recovered transition $(S \rightarrow R)$ to:

- If: RAND $< (S_S/N_S) * \nu * \Delta t$, then change state of susceptible node to recovered.

However, this change is only applied to nodes within the smaller population subset (nodes outside the subset are not being patched) where S_S is the number of susceptible nodes in the population subset at the time step and N_S is the size of the population

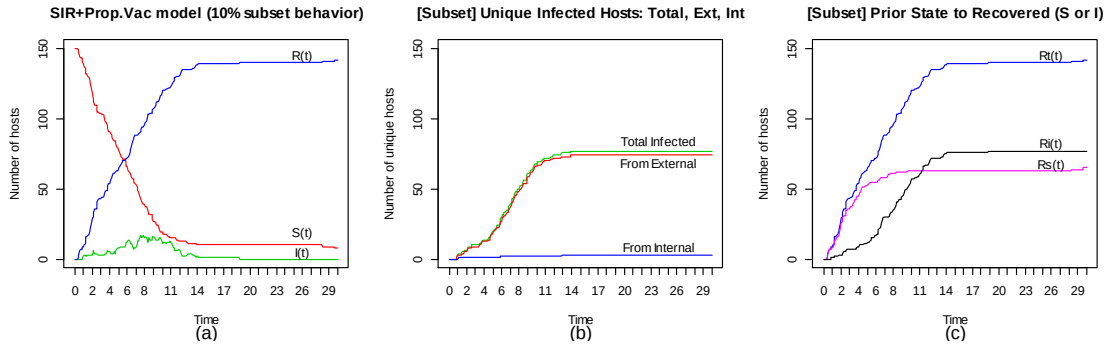

Figure 3.10: Example of proportional vaccination rate in population subset.

Table 3.4: Comparison of fixed and proportional rate patching efforts.

Population	Subpop	β	γ	Vac. Rate	Total Inf.	Sub Inf.	Sub Inf. Time
1,000	100	1.8	0.75	0.0	853	85.6	104±16.2
1,000	100	1.8	0.75	0.1 (fixed)	803	46.9	56.6±12.5
1,000	100	1.8	0.75	0.1 (proportional)	813	56.2	72.0±14.8
10,000	1,000	1.8	0.75	0.0	8,524	852	1,057±46
10,000	1,000	1.8	0.75	0.1 (fixed)	7,761	345	423±66
10,000	1,000	1.8	0.75	0.1 (proportional)	7,949	465	572±63

subset.

Figure 3.10 shows an example of patching vulnerable systems (of the population subset only) at a rate proportional to the number of susceptible systems. Table 3.4 compares some of the outcomes of proportional and fixed rate patching efforts. For the same parameter value, the proportional patching efforts result in somewhat higher infection rates. This is because the effective patching rate declines as more systems are patched rather than remaining constant until all susceptible systems are patched.

3.5 Model Application Illustration

Using models to better capture and understand the underlying dynamics affecting the proliferation of computer security incidents provides a means for identifying,

testing, and evaluating interventions designed to reduce the number of incidents. The effect of different interventions can be factored into model simulations by adjusting relevant model parameters and results for different interventions can be compared. Cost information can also be combined with model results and intervention choices to provide additional context for decision making.

3.5.1 Baseline scenario

First we define a baseline scenario that will be used as a starting point for simulation comparisons. We examine the number of infected hosts in a population of 10,000 initially susceptible hosts for a period of 60 days for 1,000 simulation runs with parameter values for infection, vaccination, and recovery rates sampled from uniform random distributions within specified ranges. For some interventions, a fixed parameter value for a blocking rate is also used. Table 3.5 provides a summary of the parameter descriptions, designations, and values used for the baseline. The next section describes the different interventions and the impact each intervention may have on the baseline parameter values.

3.5.2 Interventions

Next we describe several different types of interventions that could be employed to control types of computer security incidents for which SIR models apply. These interventions include the installation of IDS/IPS, implementation of large-scale patch management, addition of human security analyst, or some combination of these

Table 3.5: Properties of baseline case for comparisons of interventions.

Description	Designation	Value
# of nodes/computers	P (for population)	10,000
Time range	T	60 (days)
Infection rate	β	unif. random [0.9, 2.4]
Recovery rate	γ	unif. random [0.4, 0.8]
Patch/vaccination rate	ν	unif. random [0.015, 0.035]
Blocking rate	ρ	0 (no blocking)
# of simulations	N	1,000

options.

- IDS/IPS–this intervention works by monitoring network traffic between hosts and then blocking identified suspected malicious traffic. Typically identification of malicious traffic is based on some kind of pattern or signature matching. Although it will vary depending on the specific exploit being used, this type of pattern matching usually means there will be some trade-off between the number of false positives and false negatives. A very specific signature is more likely to reduce the number of false positives (i.e. the blocking of traffic that is not malicious), while also increasing the number of false negatives (i.e. the permitting of traffic that is malicious). There are both free, open-source and commercial IDS/IPS options available. Typically, one of the differences between free and commercial offerings is the time for signatures to become available for detecting new threats as well as the quality of new initial signatures. For this illustration, we assume a commercial IDS/IPS releases signatures for new

threats one day after discovery and signatures block 80% of associated malicious traffic without false positives. For comparison purposes, we assume a free IDS/IPS releases signatures for new threats three days after discovery and signatures block 70% of associated malicious traffic without false positives.

- Patch management–this intervention works by automating and/or simplifying the process of installing updates and patches to fix known vulnerabilities. This can increase the rate at which susceptible hosts are removed and directly transition to the recovered state without first becoming infected. Patches may be for vulnerabilities found in operating systems and/or installed applications. The complexity of a patch management solution may be very dependent on the homogeneity or heterogeneity of installed operating systems and applications in a large population.

- Security analyst–this intervention works by adding human analyst to monitor, review, evaluate reports of events and potential malicious activity. The analyst has the ability to isolate or block hosts which are deemed to be threats to other hosts. An analyst will also review attack trends and will actively look for early signs of malicious activity. Because of this, we assume a decrease from two days to one day for the time gap between infection and discovery of initial incidents. We also assume an analyst will increase the general rate infected hosts are detected and removed/recovered.

Table 3.6 summarizes the intervention options and identifies some of the combinations of interventions explored. When applicable, differences from the

Table 3.6: Intervention properties.

Index	Description	Cost ($)	Delay ($S \rightarrow I$)	Block delay	Block rate	Patch rate	Recov. rate
1	Baseline	0	2	-	0	-	-
2	Free IDS (fIDS)	10k	2	3	0.7	-	-
3	Pay IDS (pIDS)	55k	2	1	0.8	-	-
4	Patch mgt (PM)	30k	2	-	0	+0.025	-
5	Analyst	75k	1	-	0	-	+0.15
6	fIDS & PM	40k	2	3	0.7	+0.025	-
7	pIDS & PM	85k	2	1	0.8	+0.025	-
8	fIDS & Analyst	85k	1	3	0.7	-	+0.15

baseline scenario parameters are indicated. Also included in the table is an estimated cost value for an intervention or combination of interventions. The options shown satisfy a cost constraint that intervention implementations must not exceed $100,000 ($100k). This constraint was imposed as an example of a constraint due to finite resources. Cost aspects are further discussed in the next section.

3.5.3 Costs related to outcomes and interventions

Two cost aspects considered when comparing interventions are the cost of the intervention itself as well as the costs related to the outcomes. The outcome costs include estimated costs for recovering or restoring infected hosts (per host), estimated costs for patching susceptible hosts (per host), and an estimated value related to increased risk of other types of malicious activity from infected hosts during the time they are infected. Table 3.6 shows values for each intervention option explored

Table 3.7: Outcome related costs.

Description	Cost ($)
Patched node (includes risk of bad patch)	1
Infected → Recovered node	5
Max. exposure (total time infected) risk cost	10,000

and Table 3.7 shows recovery and patching costs (per host) as well as the maximum value attributable to the risk of infected hosts being used for other types of malicious activity.

The associated risk cost is calculated using a logistic (or sigmoid) function based on the simulation results of the baseline case. The total time infected across hosts in a simulation is divided by the maximum total time infected across all hosts using the baseline scenario (which was 21,673.8 "host-days"; the theoretical maximum is 10,000 nodes * 60 days = 600,000 "host-days"), then an associated risk cost value is determined. The curve centered around 12,500, which means if a simulation results in 12,500 total time hosts are infected, the associated risk cost value is $5,000 (or 1/2 of the maximum $10,000 cost). The cost value for patching hosts is intended to include both the time and effort to test and apply patches, but also additional costs that may be incurred if a patch does not work as intended and causes downtime. The cost value for recovering nodes is intended to include the time and effort to restore original functionality to a host while also ensuring it is no longer susceptible. These cost values are likely to be organization dependent and the values shown are for illustrative purposes only.

Figure 3.11 shows the implementation costs for each intervention (labeled as "Base costs") as well as box plots for range of patching, recovery, and risk costs for 1,000 simulations for each intervention option. Figure 3.12 shows box plots for all of the outcome related costs (patching, recovery, and risk) combined while Figure 3.13 also adds in the implementation costs for each option to show the range of total costs. While total costs may be one of the factors considered when comparing intervention options, reduction of uncertainty may be another factor used during evaluation of choices.

Looking at the total costs shown in Figure 3.13, intervention option 2 (free IDS) has the lowest median cost based on simulations. Intervention option 3 (commercial IDS) has the fourth highest median cost based on simulations, but the least variability among the five lowest median cost interventions.

In this chapter, we described and illustrated a method for modeling the spread of computer security incidents based on the SIR compartmental type of model originally applied to the spread of some types of diseases in human and animal populations. We demonstrated how different types of typical intervention actions (blocking and patching) could also be represented using the model. We illustrated how the model could be modified to examine behavior within a particular subset of the population and explained why this would be a desirable property. Using the models, we illustrated how the time-to-first-infection in the population subset decreased as the size of the population subset increased relative to the size of the full population. We also explored and illustrated how a proportional rather than fixed rate of patching allows for an increased number of infections to occur because

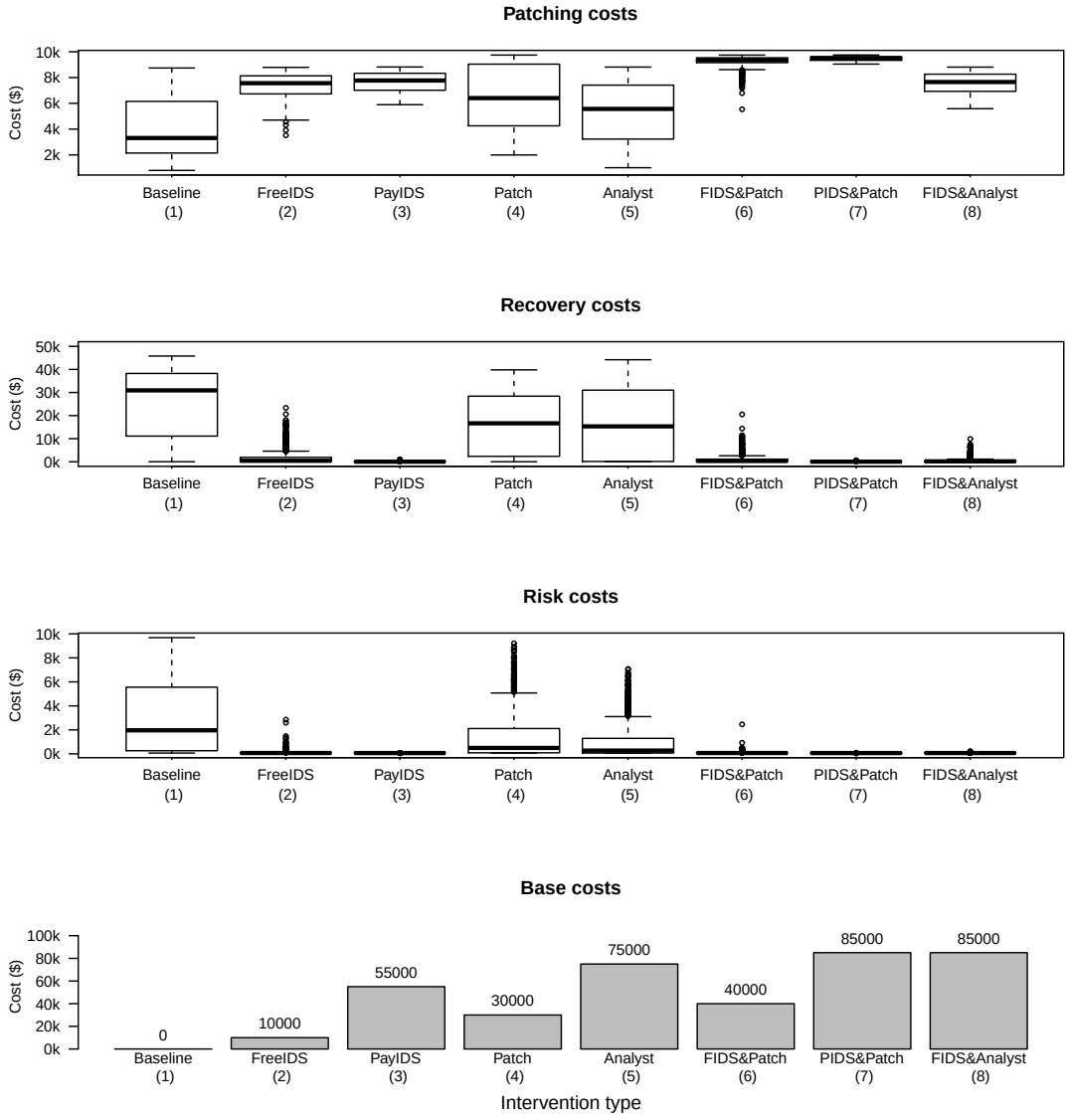

Figure 3.11: Intervention cost components.

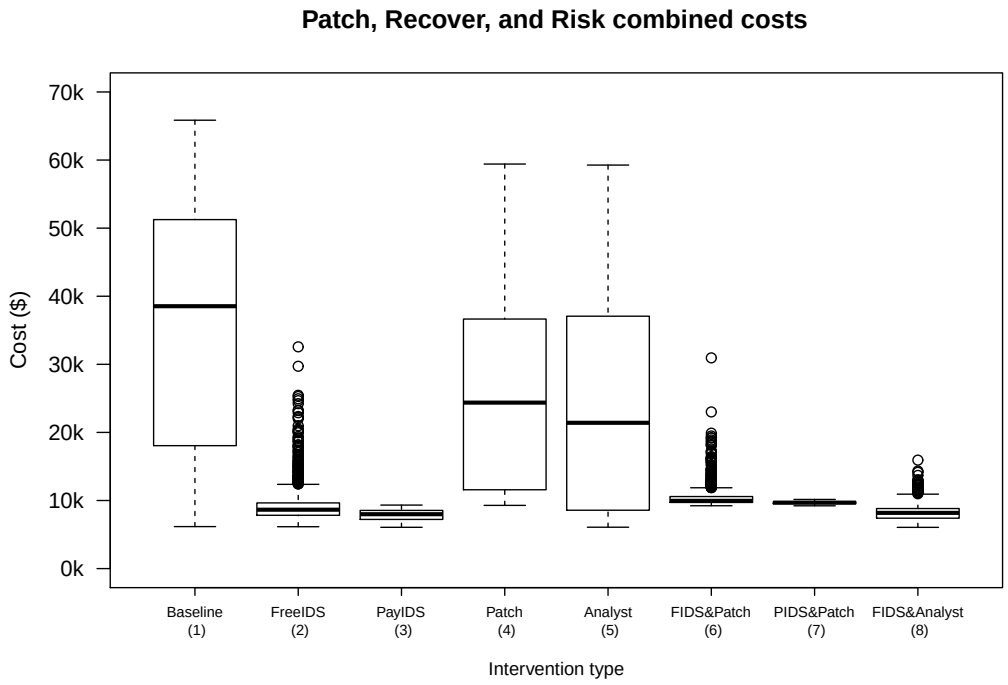

Figure 3.12: Intervention options outcome costs combined.

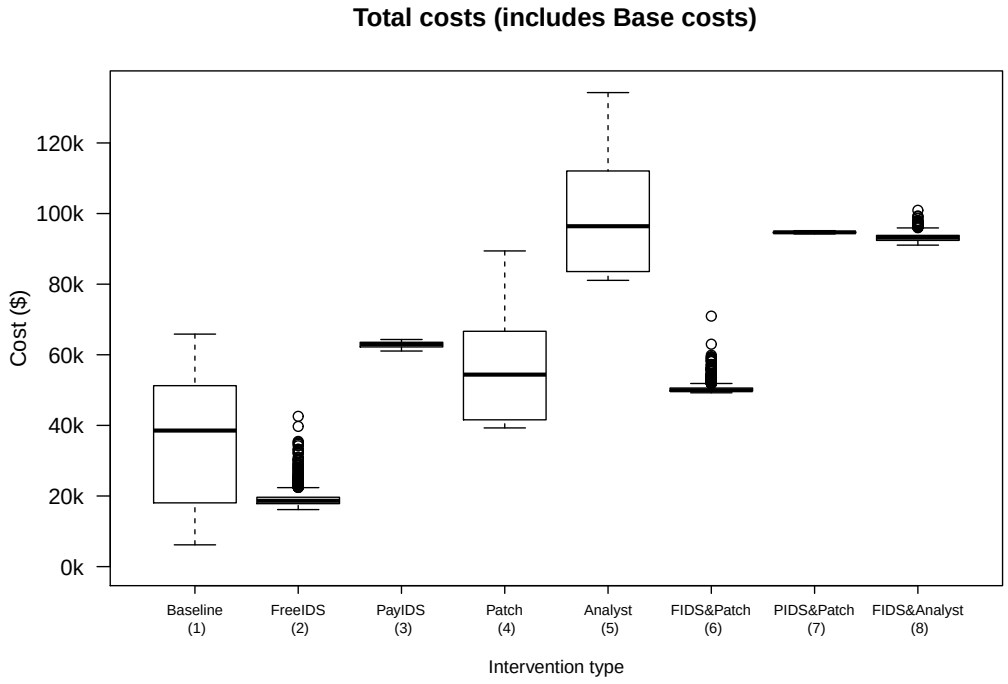

Figure 3.13: Intervention options total combined costs (implementation and outcomes).

of a decreased rate of patching over time as the number of remaining unpatched computers also decreased. We provided an example illustrating the use of the models (in combination with cost information) to evaluate and compare outcomes resulting from different intervention decisions. This demonstrated a method for using the models as a resource planning tool. The example showed how the different types of cost information could be combined with the model outcomes to provide final costs estimates which take into account intervention and outcomes costs. Several types of intervention implementations were modeled and compared. For the values used in the illustration, we saw that while total costs for some interventions were higher than others, these intervention actions may also result in less variable outcomes which could be a desirable aspect.

Chapter 4: Email Propagation Models

4.1 Overview

In this chapter, we describe a stochastic model for a worm or virus that spreads using email, and we illustrate some aspects of the model in an interconnected population of computers and users. This model is similar to the stochastic compartmental SIR model covered previously, but there are some important differences. For some transitions to occur in the email propagation model, several conditions may be required. While the previous SIR model assumes direct contact between computers and does not include aspects of a user, the email propagation model includes two aspects related to user interaction. These are the frequency at which email messages are checked and the user likelihood for opening an infected email message. An overview of the basic stochastic email propagation model is followed by more discussion and illustration of some aspects of modeling and the use of interventions for controlling the spread of computer security incidents that transfer via email.

4.2 Stochastic Email Propagation Model

Like the previously described SIR models, the email propagation model designates hosts to be in one of three compartments or states: Susceptible (S), Infected (I), or Removed/Recovered (R). However, the transition from $S \rightarrow I$ is defined differently and requires more than one condition to be met before a transition will occur. A summary of the steps taken to determine if a susceptible node transitions to an infected node is as follows:

- For each susceptible node, identify already infected neighbors. Neighbors are other nodes with a direct connection to the susceptible node. Different connection topologies can be used, such as fully-connected, randomly connected, or scale-free (power law).

- For each susceptible node, determine the number of new infected messages received. A binomial distribution, based on the number of infected neighbors, is used for determining the number of new infected messages sent.

- For each susceptible node, add the number of new infected messages received to the number of previously unchecked infected messages.

- For each susceptible node, determine if the associated user (assuming a one-to-one correspondence between number of nodes and number of users) checks messages. At each time step, a uniform random probability distribution is used to determine if a user has checked messages. For a large number of users, this results in a approximation of a normal distribution.

- If a user checks messages, then for each infected message present, determine if the user has opened the infected message. Each user is assigned a value representing the likelihood of opening an infected message. The assigned values are based on an exponential distribution where most users have a low likelihood of opening an infected message, but a very small number of users have a higher likelihood of opening an infected message. If one or more infected messages are opened, then that node changes from susceptible to infected.

The transition from $I \rightarrow R$ is the same as in the SIR models described previously and depends on the recovery rate parameter γ. The transition from $S \rightarrow R$ is also the same as in the previous models with ν as the vaccination rate; however, it should be noted that the node becomes vaccinated, not the associated user. This means a recovered node may still receive infected messages and the associated user may still open the infected messages, but the node itself will no longer become infected and will not send out infected messages to its neighbors. The recovery and vaccination rate parameters are assumed to remain relatively constant or fixed in time for the interval being modeled. Some of the previously described steps are similar to steps described in [70]. However, our models are an entirely independent implementation using the R [68] programming language and include variations not previously examined. An example of the R code used for the email models is provided in Appendix C.

4.3 Network Topology

The SIR models covered in Chapter 3 assume fully-connected networks. This may be a reasonable assumption for worms that propagate directly from computer-to-computer; however, malware that spreads via email may not fit with this assumption. Email propagation involves a user component which is not present in the previously described SIR models. While the computers themselves may be fully connected through the same network links used by worms that spread directly from computer-to-computer, the spread of malware through the exchange of email messages is more likely to follow or be influenced by connections or contacts between users instead of the network links between computers.

Scale-free (or power law) networks [69] are one type of connection topology that have been used to model social connections and may be suitable for representing email exchanges. Propagation in scale-free networks can be compared to propagation in randomly connected networks that use approximately the same number of links but have a different node-degree distribution. Figure 4.1 shows some example network connection topologies and their associated frequency distribution of node-degrees. While the fully-connected example with 10 nodes has a total of 45 connection links, both the scale-free and randomly-connected examples have a total of 24 connection links each. Although these examples only include 10 nodes, the node degree histogram for the scale-free network is consistent with a power law distribution while the node degree histogram for the randomly connected network is consistent with a normal distribution.

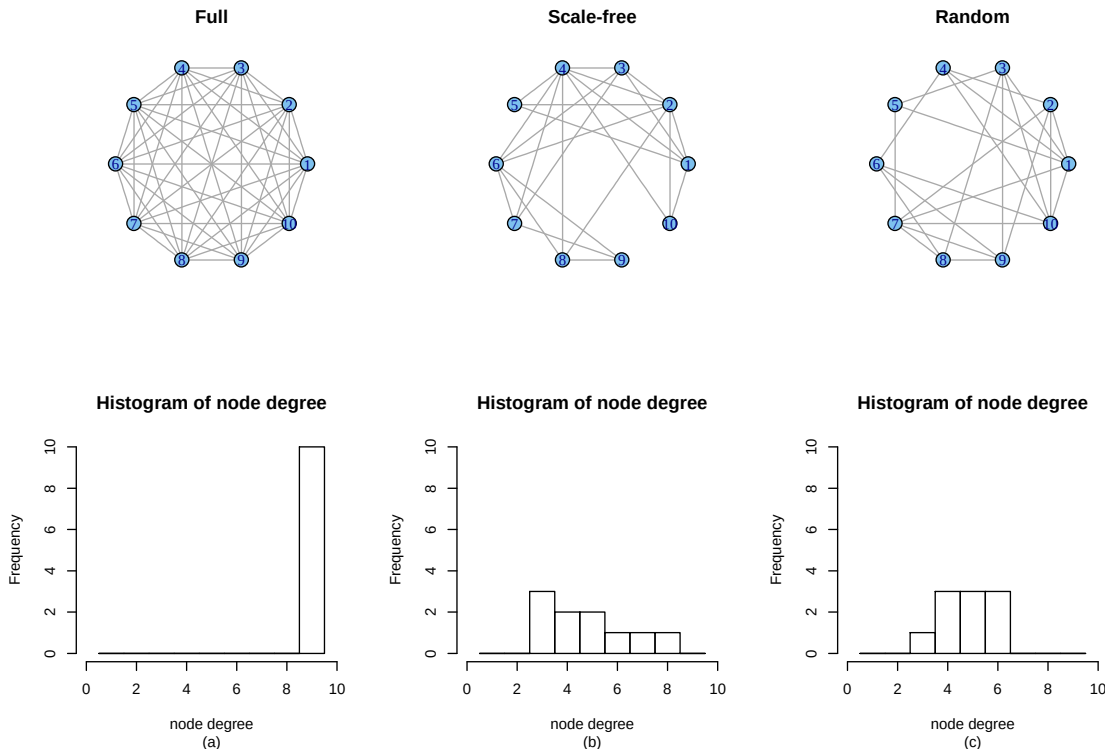

Figure 4.1: Network connection topology examples and node-degrees.

The arrangement of links in the network may influence the spread of an email virus, but the overall level of connectedness may also have an impact. There is more potential for an email virus to spread in a more fully connected network than in a sparsely connected one. This aspect is explored and illustrated later.

Figure 4.2 shows an example of number of nodes in each state over time for a scale-free network topology example. Also shown is the number of infected messages sent at each time step along with a histogram of the node-degree distribution of the network. Figure 4.3 shows the same information, but for a randomly connected network topology example for the same number of nodes and approximately the same number of links used in the scale-free example. Although a similar number of nodes become infected in each example, the time at which they become infected, as

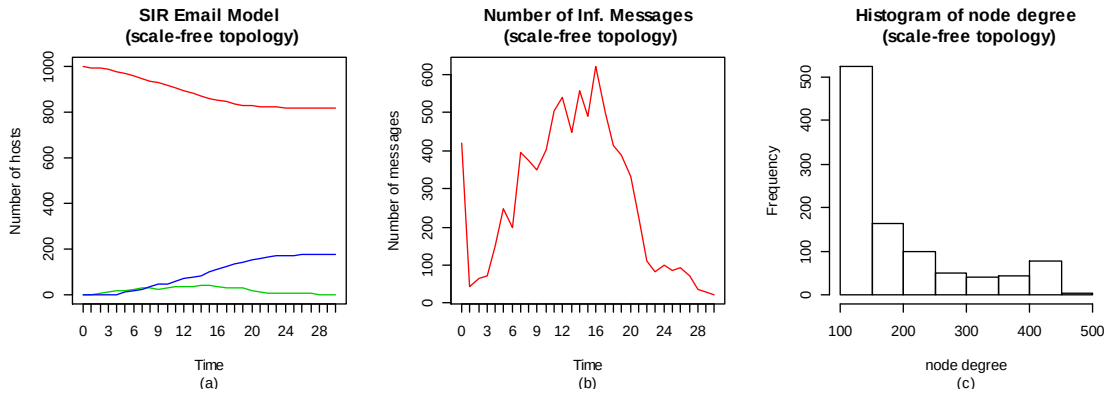

Figure 4.2: Scale-free network topology example (≈20% connected, $\beta = 1.0$, $\gamma = 0.3$).

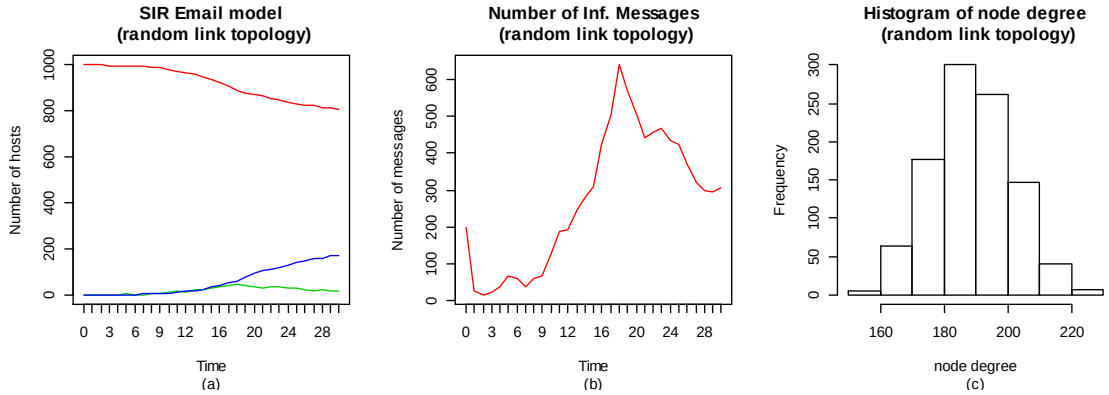

Figure 4.3: Random network topology example (≈20% connected, $\beta = 1.0$, $\gamma = 0.3$).

well as when many of the infected messages are sent, is different in each case. In these examples, the infection appears to take longer before it begins to spread in the randomly connected network than in the scale-free connected network. Table 4.1 shows some of the values used and obtained for the examples illustrated in Figures 4.2 and 4.3.

Table 4.1: Parameters and outputs for topology examples shown in Figures 4.2 and 4.3

Topology	Number of nodes	Number of links	Mean node degree	Total nodes infected	Total time infected
Scale-free	1,000	94,950	190	181	580
Random	1,000	94,758	190	191	624

4.4 Human Factors

The previously covered SIR models do not include aspects of human user activity on the infectious process. Those models assume that human action is not necessary for a worm or virus to spread as an infection is passed directly from computer-to-computer though a network connection. However, a model for how an infection may spread through email does incorporate elements of human users. We describe two primary aspects included in our models below.

4.4.1 Email checking interval rate

A network worm or virus that spreads directly from computer-to-computer without requiring any user interaction can be approximated as a continuous process. However, in the case of a worm or virus that spreads through email and requires some kind of user interaction (such as opening an infected message), the spreading process may be more discrete or event-driven in nature.

Because user interaction is required for an email worm or virus to spread, each node in the model or simulation has an associated human user. One aspect of the user included in the model is how often a user checks or reads new messages. For each time step, the base model selects a value from a uniform random distribution (between 0 and 1) for each user; if the value exceeds a threshold value (presented simulations use a threshold value of 0.5), the user is defined as having checked for messages at that time step. This results in an overall normal distribution for frequency of message checking across all users.

The above technique for modeling user email checking behavior treats all users the same and samples from a uniform random distribution. However, other variations could also be used. One modification could be to adjust the threshold value in relation to the size of the time step so that the mean value is centered around a desired target value. For example, to represent users having a 70% daily rate for email checking, a threshold value of 0.70 could be used when one time step is equal to one day, or a threshold value of 0.07 could be used when one time step is equal to 1/10 of a day. This would allow for the same mean value of email checks during the same time interval (one day).

Another modification could be to sample from a different type of distribution, such as a normal or an exponential distribution. An example of this is presented in [70]. Another approach could be to model each user independently by assigning each user a parameter value used for defining individual user-specific distributions to sample from. This would capture the possibility that some users may consistently check messages more frequently than others, and this behavior, when coupled with a higher number of contacts, could change the spreading behavior of a worm or virus. These variations are not explored in this document.

Figures 4.4 through 4.7 show examples of changing the threshold value used for simulating the email checking frequency of the model. Figure 4.4 corresponds to a user having a 25% chance of checking for new email messages at each time step. Figures 4.5, 4.6, and 4.7 correspond to 50%, 75%, and 100% chances of a user checking for new email messages at each time step respectively.

Figure 4.4 shows an example where the user population averages 7.5 total

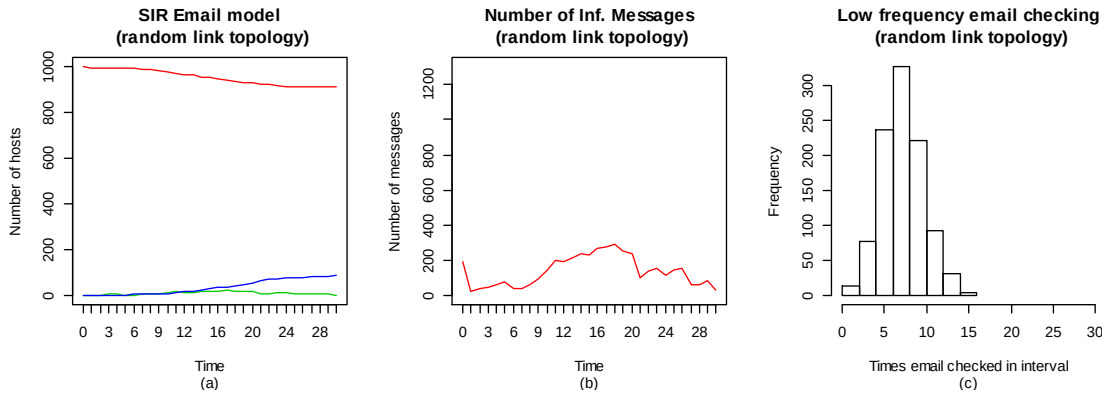

Figure 4.4: Random topology, less frequent email checking.

times checking for emails during the 30-day simulation run (as seen in Figure 4.4(c)). Figure 4.4(a) shows the lowest number of infected nodes during the time span compared to Figures 4.5(a), 4.6(a), and 4.7(a). Figure 4.4(b) shows the lowest number of infected messages being sent overall and at peak value compared to Figures 4.5(b), 4.6(b), and 4.7(b). Since emails are checked less frequently, there are fewer opportunities for the worm or virus to spread. Since the detection and removal rate parameter is unchanged, infections that do occur are still removed at the same rate. A slower infection rate combined with an unchanged removal rate results in fewer overall infections during the same time window and likely for an extended window as well. Increased email checking by the associated user population appears to push the timing of peak infection occurrence and the timing of peak infected messages sent earlier in the time interval; the magnitude of these peaks appear be larger as well. Table 4.2 indicates that similar observations likely apply to scale-free connected topologies, as the total number of infected nodes and the total time of nodes in the infected state increase as the rate of email checking increases.

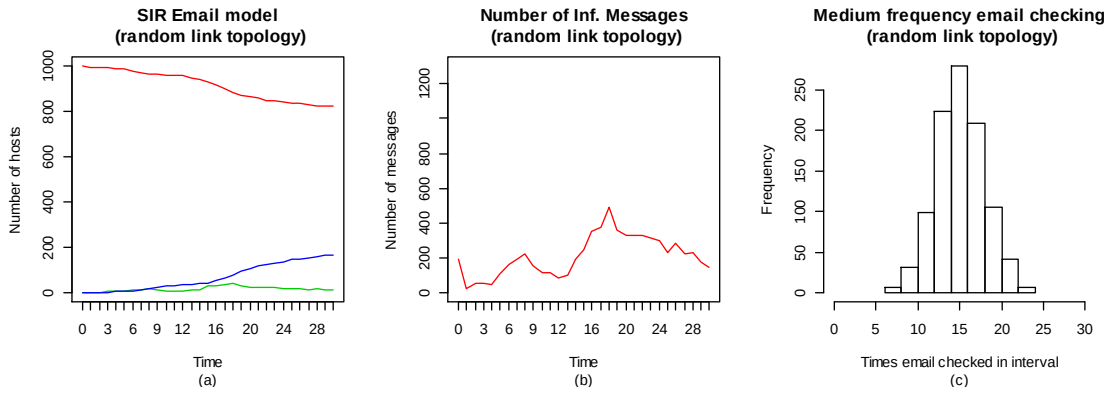

Figure 4.5: Random topology, medium frequent email checking.

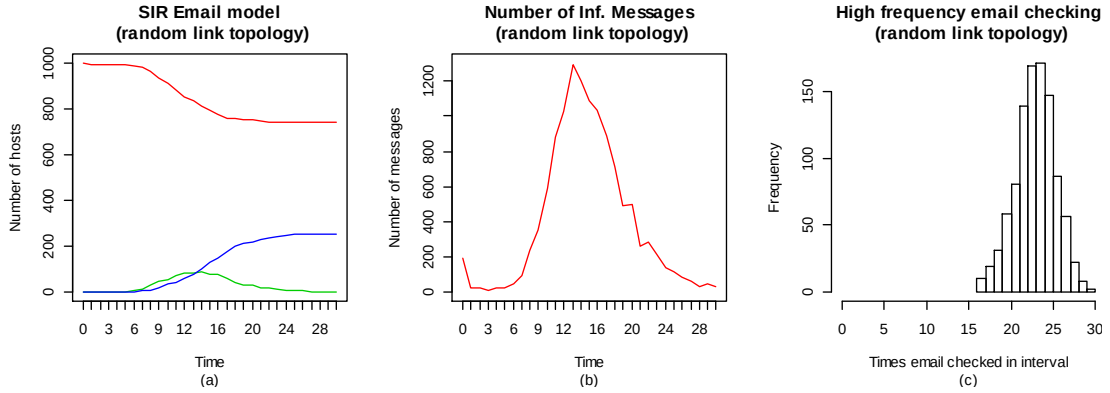

Figure 4.6: Random topology, more frequent email checking.

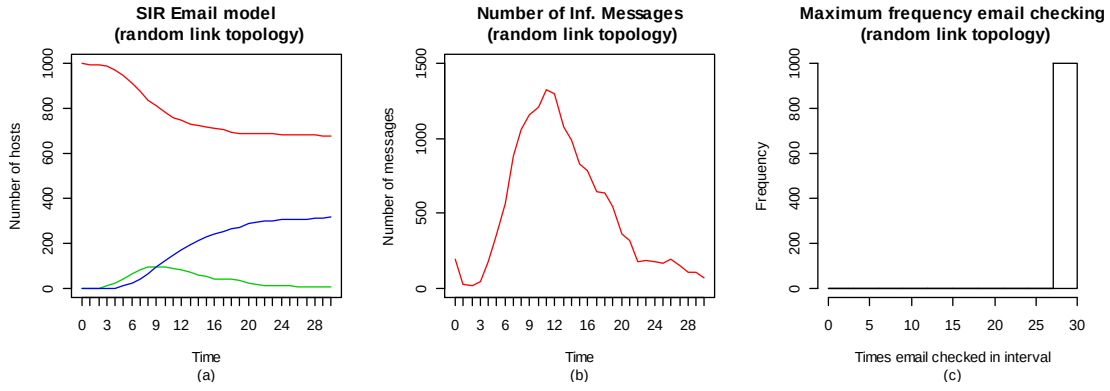

Figure 4.7: Random topology, maximum email checking.

Table 4.2: Examples of different email checking intervals on number of infections (nodes = 1,000)

Topology	Number of links	Number reps > 10	Frequency of checking	Total nodes infected	Total time infected
Random	95,067±262	30	Low	54±38	180±141
Random	94,940±307	70	Medium	212±34	690±140
Random	94,957±302	83	High	293±16	969±79
Scale-free	94,950	50	Low	62±33	193±113
Scale-free	94,950	85	Medium	207±51	677±186
Scale-free	94,950	95	High	285±14	949±84

4.4.2 Likelihood to open infected message

Another user aspect included in the model is the likelihood that a user will open an infected message. For modeling purposes, each user has an assigned likelihood threshold value for opening an infected message. This value is compared to sampled values to determine if a particular user opens a particular infected message. The base model samples from a uniform random distribution (between 0 and 1). Then for each infected email message, the sampled value is compared to the predefined threshold value (also between 0 and 1)to determine if the infected message is opened. If the sampled value is less than the threshold value for a particular user, the model assumes the infected message was opened and an infection may occur. The assigned threshold values remain unchanged for each user throughout a simulation.

The overall distribution of assigned likelihood (or threshold) values for all users is based on an exponential distribution where the majority of users have low likelihood values (i.e. these users are less likely to open an infected message), but a few users have higher likelihood values. We explore the use of exponentially based distributions for assigning likelihood values because it has previously been observed

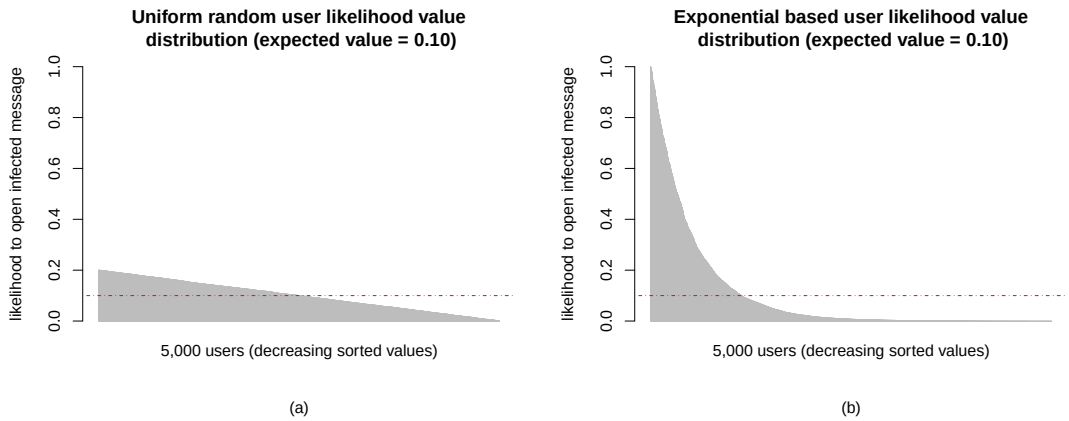

Figure 4.8: User likelihood distribution examples with similar expected values. *[Red line shows the population expected value.]*

that in many types of tasks, the distribution of the number of accidents among people is better described by a Poisson distribution than a Gaussian distribution [71]. The opening of infected email messages may follow a similar distribution to these accidents. Other distributions for assigning user likelihood values for opening infected messages are possible and we make comparisons between models using the uniform random distribution for assigning user likelihood threshold values. The work presented in [70] only illustrates using a Gaussian distribution for modeling this factor.

When making comparisons between models using different types of distributions for assigning user likelihood threshold values, we attempt to keep the expected values that an infected message would be opened the same for the populations (see Figure 4.8). This means, if other factors are equal, about the same number of infected messages would be opened, but the distribution of opened messages among users (i.e. 10 different users opening one infected message each versus 1 user opening 10 infected messages alone) of opened infected messages would be different when assigning user likelihood values using different types of distributions.

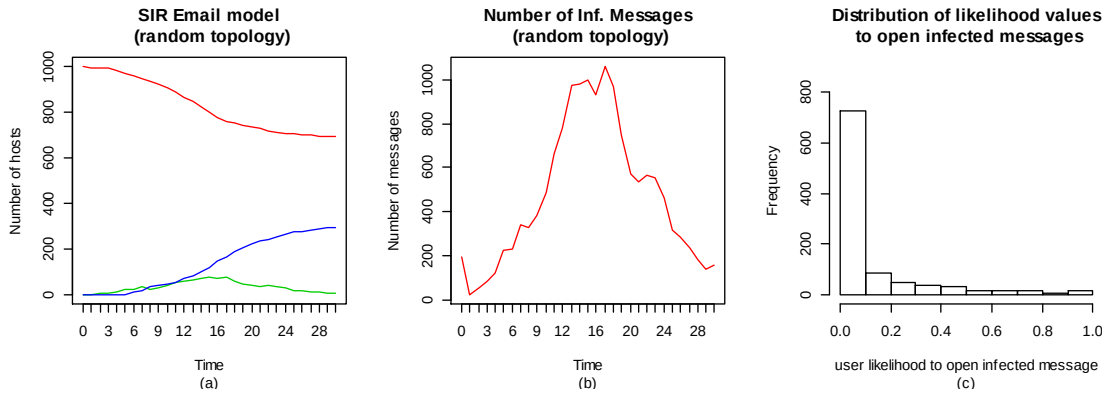

Figure 4.9: Random topology (≈20% connected), expected user likelihood for opening infected messages ≈12.5%, exponential distribution of likelihoods.

The user likelihood values for opening infected messages are assigned independently of the associated node connection topology. There is likely an interaction effect between these two factors. For example, a highly connected node with an associated user that is assigned a high likelihood for opening infected messages may extend the spread of a worm or virus that spreads through email. Because the node is highly connected, it has a higher chance of receiving infected messages due to having a higher number of neighbors. Since the associated user is also more likely to open an infected message (thus infecting the node), the node also has the potential to send out a larger number of infected messages due to being highly connected.

The comparison of Figure 4.9 to Figure 4.10 and Figure 4.11 to Figure 4.12 shows an increase in the overall number of infections when a higher number of users is more likely to open an infected message (regardless of the specific network topology) for an exponentially based distribution of likelihood values. The magnitude of the change appears to be larger for the random network topology, but Figures 4.9-4.12 are single simulation examples, and there may be other sources of variability.

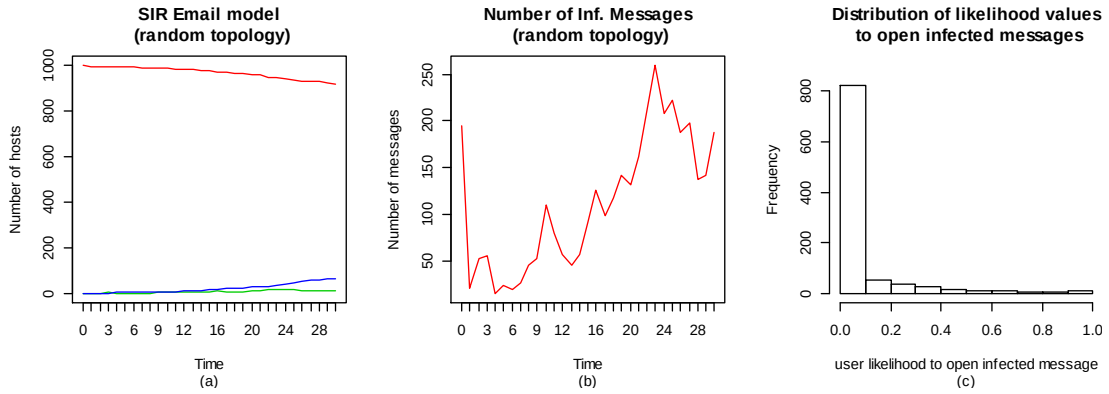

Figure 4.10: Random Topology (≈20% connected), expected user likelihood for opening infected messages ≈8.33%, exponential distribution of likelihoods.

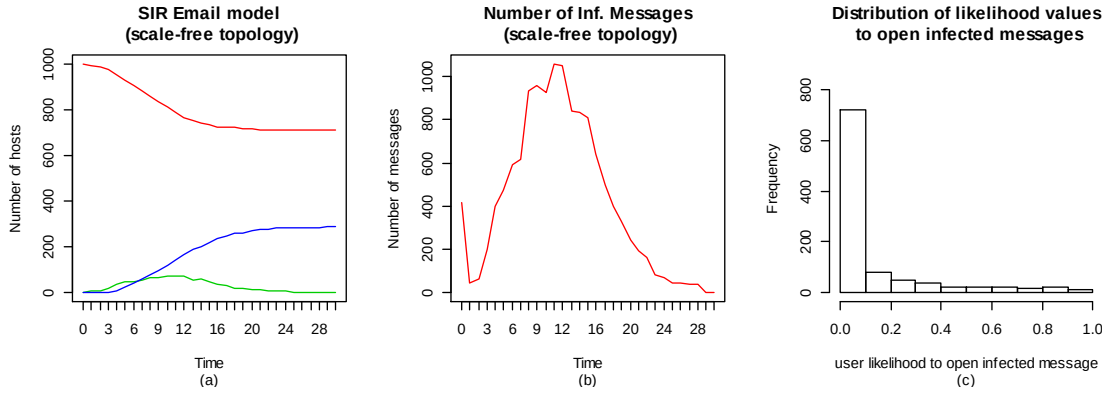

Figure 4.11: Scale-free topology (≈20% connected), expected user likelihood for opening infected messages ≈12.5%, exponential distribution of likelihoods.

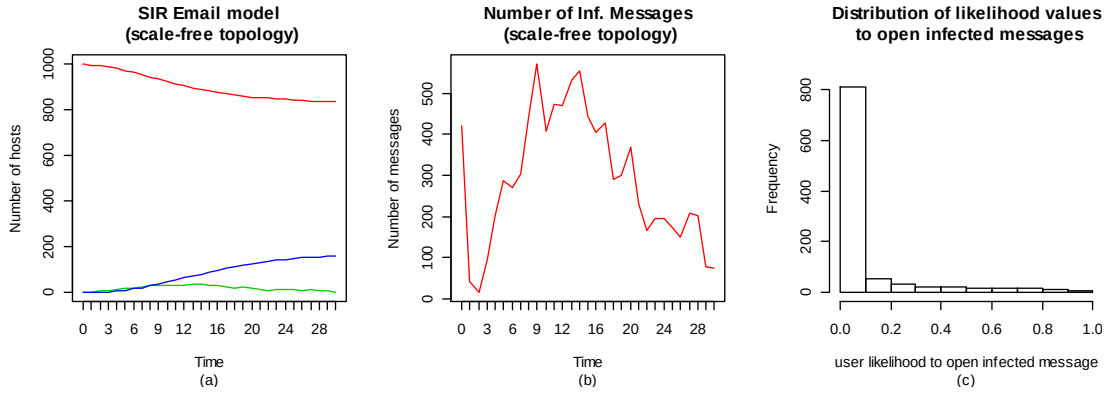

Figure 4.12: Scale-free topology (≈20% connected), expected user likelihood for opening infected messages ≈8.33%, exponential distribution of likelihoods.

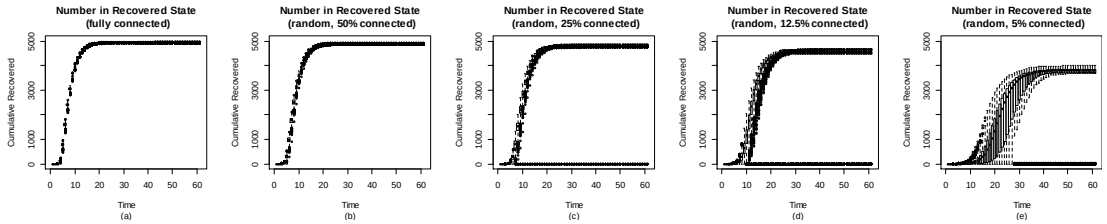

Figure 4.13: Uniform random likelihood to open infected messages; randomly connected network with different link levels.

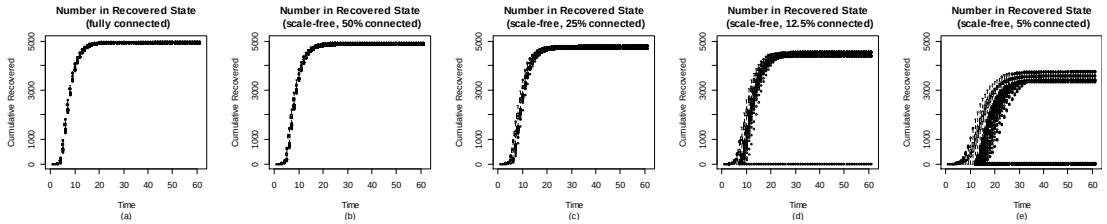

Figure 4.14: Uniform random likelihood to open infected messages; scale-free connected network with different link levels.

Figures 4.13-4.16 show boxplots summarizing the number of recovered nodes for 2,500 simulations using 5,000 nodes for both random and scale-free network topologies at several different connectivity levels. Figures 4.13 and 4.14 show the results of using a uniform random distribution of values for user likelihoods for opening infected messages. Figures 4.15 and 4.16 show the results of using an exponentially based distribution of values with the same overall expected value as the uniform random distribution used in Figures 4.13 and 4.14. More variability in the results is seen in the randomly connected topology examples Figures 4.13 and 4.15) compared to the scale-free examples (Figures 4.14 and 4.16). In particular, instances of non-spreading infections appear in some of the 25% connectivity level examples (Figures 4.13(c) and 4.15(c)), while non-spreading infections first appear in the 12.5% connectivity level scale-free examples (Figures 4.14(d) and 4.16(d)).

Variability also increases as the level of connectivity decreases. For both types

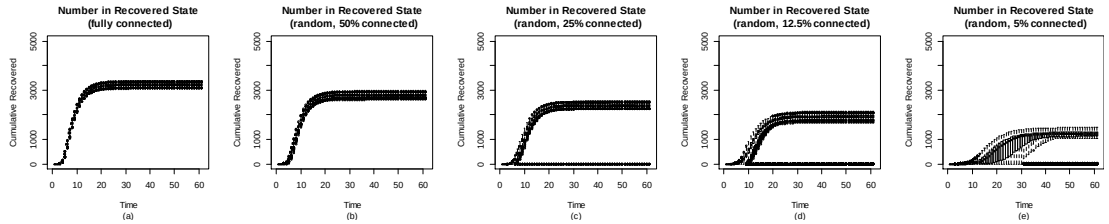

Figure 4.15: Exponential based likelihood to open infected messages; randomly connected network with different link levels.

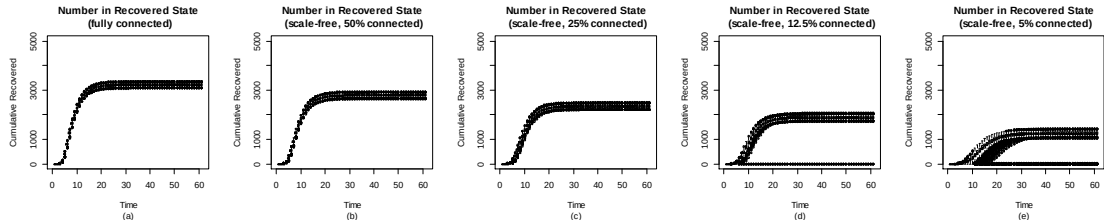

Figure 4.16: Exponential based likelihood to open infected messages; scale-free connected network with different link levels.

of network connection topologies, the exponentially based distribution of values for user likelihoods for opening infected messages results in fewer overall infected (or recovered) nodes for the same overall expected value of infected messages opened.

4.5 Interventions (Blocking and Patching)

The two interventions discussed in regard to the SIR model in Chapter 3 can also be applied to the email propagation model–blocking and patching. However, instead of blocking at a network "border" as discussed for the previous SIR models, blocking could occur at a mail server that receives/delivers messages for the set or subset of users associated with the nodes or hosts. Messages being sent from one node to another node will traverse a mail server which can be set to detect and block transmission of infected messages. The implementation for the blocking (or filtering) of messages may not happen right away. In cases of a delay, it is possible

for unopened infected messages to exist (and potentially cause new infections) even after blocking is implemented; however, further spreading beyond the newly infected node(s) would be blocked.

Patching underlying vulnerabilities is applied in the same way as in the previous SIR models in Chapter 3. Patching is applied to nodes or hosts. It does not prevent infected messages from being received or opened by a user, but it would prevent a node or host from becoming infected if a user were to open an infected message.

By including human user aspects in the email propagation model, an additional intervention can be explored. Because a user needs to open an infected message (on a susceptible host) in order for an infection to occur, user education or user-awareness training can try to reduce the likelihood users will open infected messages. As previously mentioned, the impact of changing a user's likelihood for opening an infected message will partially depend on the number of contacts for the user (reflected in the connectedness of the associated node).

Figures 4.17-4.18 show examples of the results of applying patching, blocking, and a combination of both interventions for randomly and scale-free connected network topologies at different connectivity levels. The randomly connected topology examples use a uniform random distribution for the user likelihood values for opening infected messages, while the scale-free examples use an exponentially based distribution (with the same overall expected value) for these values. Shown are boxplots of the number of recovered nodes for 2,500 simulations using 5,000 nodes. For examples where the patching intervention is used, the boxplots show only the number of recovered nodes that transitioned from the infected state and do not

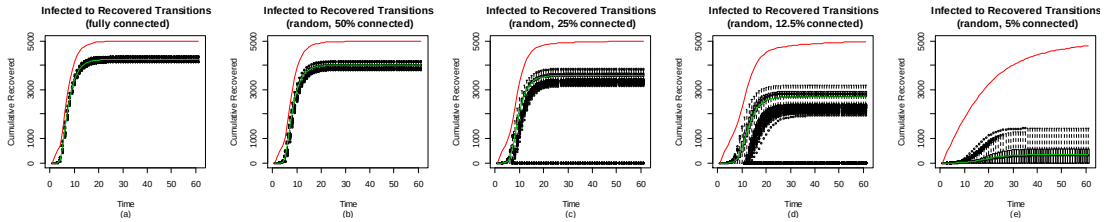

Figure 4.17: Effect of patching vulnerable computers; random topology, unif. random likelihood for opening infected emails. *[Red line is mean value for number of patched nodes for reference, green line is mean value for nodes that become infected before being recovered.]*

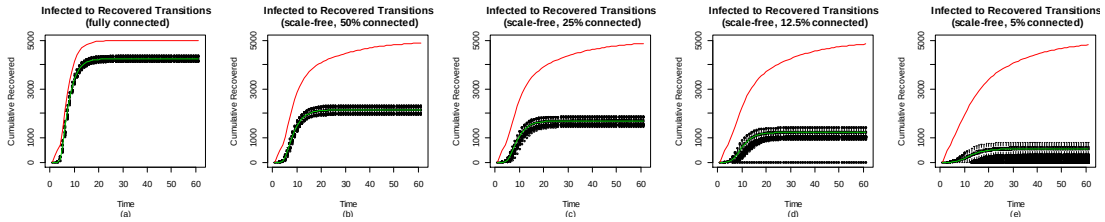

Figure 4.18: Effect of patching vulnerable computers; scale-free topology, exp. based likelihood for opening infected emails. *[Red line is mean value for number of patched nodes for reference, green line is mean value for nodes that become infected before being recovered.]*

include nodes that were patched (and transitioned directed from the susceptible state to the recovered state). The red lines in Figures 4.17 and 4.18 provide a reference for the mean value of the number of nodes that become recovered by being patched. The red lines in Figures 4.19-4.22 show the mean value of the number of nodes in the recovered state when no interventions have been applied.

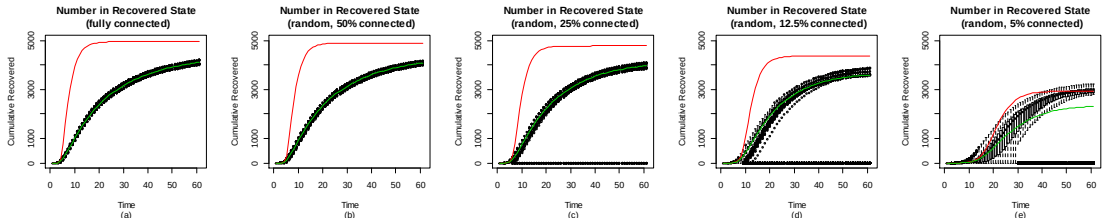

Figure 4.19: Effect of blocking most infected emails; random topology, unif. random likelihood for opening infected emails. *[Red line is mean value for unblocked simulations for reference, green line is mean value for simulations with blocking.]*

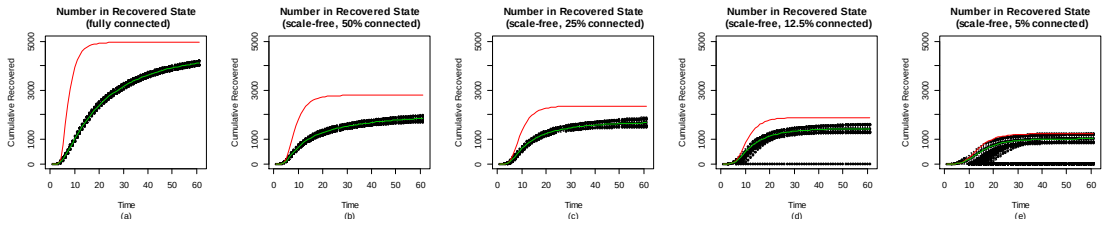

Figure 4.20: Effect of blocking most infected emails; scale-free topology; exp. based likelihood for opening infected emails. *[Red line is mean value for unblocked simulations for reference, green line is mean value for simulations with blocking.]*

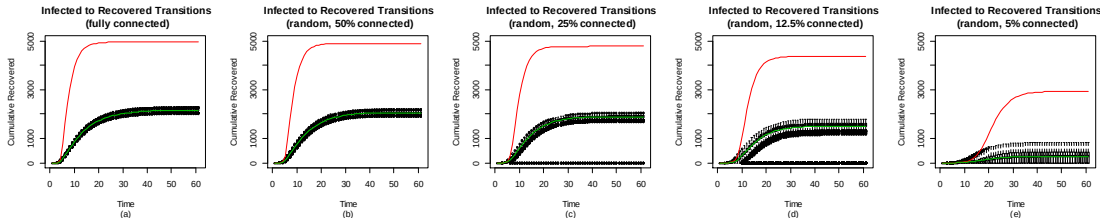

Figure 4.21: Effect of blocking and patching; random topology, unif. random likelihood for opening infected emails. *[Red line is mean value for unblocked and unpatched simulations for reference, green line is mean value for simulations with blocking and patching.]*

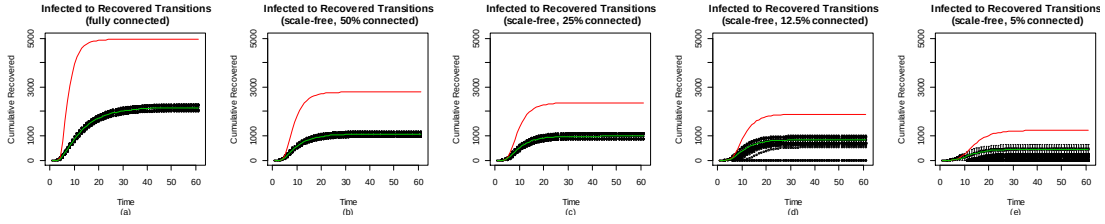

Figure 4.22: Effect of blocking and patching; scale-free toplogy; exp. based likelihood for opening infected emails. *[Red line is mean value for unblocked and unpatched simulations for reference, green line is mean value for simulations with blocking and patching.]*

The following observations regarding the simulation examples shown in Figures 4.17-4.22 can be made:

- Blocking (Figures 4.19 and 4.20) seems to slow the spread of the infection, but does not seem to reduce the total number of infections as much as patching (although this does depend on the specific parameter values used for these simulations);

- More notably, Figures 4.19(e) and 4.20(e) indicate that the effectiveness of blocking declines as the network becomes more sparsely connected;

- Comparing Figures 4.17(e) and 4.18(e) with Figures 4.19(e) and 4.20(e) respectively, indicates that patching retains its effectiveness in reducing the spread of infections better than blocking across declining levels of connection links (for the specific parameter values used for these simulations).

4.6 Model Application Illustration

This section provides illustration of using the previously described models to compare different interventions and including cost information to provide additional context for decision making.

4.6.1 Baseline scenario

First we define a baseline scenario that will be used as a starting point for simulation comparisons. We examine the number of infected hosts in a population of

Table 4.3: Properties of baseline case for comparisons of interventions.

Description	Designation	Value
# of nodes/computers	P (for population)	10,000
Connection topology	Topo	scale-free
Fraction connected nodes	FC	unif. random [0.1, 0.2]
Time range	T	90 (days)
Infection rate	β	unif. random [0.2, 1.2]
Recovery rate	γ	unif. random [0.2, 1.0]
Patch/vaccination rate	ν	unif. random [0.015, 0.035]
Blocking/filtering rate	ρ	0 (no blocking/filtering)
User likelihood for opening	UL	unif. random [0.05, 0.15]
# of simulations	N	1,000

10,000 initially susceptible hosts for a period of 90 days for 1,000 simulation runs each with parameter values for infection, vaccination, and recovery rates sampled from uniform random distributions within specified ranges. All of the simulations included in this section use a scale-free network connection topology and an exponential based distribution for user likelihood values for opening infected messages. For some interventions, a fixed parameter value for a blocking rate is also used, but no messages are blocked or filtered in the baseline scenario. Table 4.3 provides a summary of the parameter descriptions, designations, and values used for the baseline. The next section describes the different interventions and the impact each intervention may have on the baseline parameter values.

4.6.2 Interventions

Next we describe several different types of interventions that could be employed to control types of computer security incidents for which the described email propagation model applies. These interventions include the installation of mail blocking/filtering, implementation of large-scale patch management, implementation of user awareness training, or some combination of these options.

- Mail blocking/filtering–this intervention works by scanning email traffic content arriving at a mail server then blocking identified suspected malicious messages. Typically identification of malicious email content is based on some kind of pattern or signature matching of several of the email message and email header components. Although it will vary depending on the attack vector being used, this type of pattern matching usually means there will be some trade-off between the number of false positives and false negatives. A very specific signature is more likely to reduce the number of false positives (i.e. the blocking of messages that are not malicious), while also increasing the number of false negatives (i.e. the permitting of messages that are malicious). There are both free, open-source and commercial mail filtering options available. Typically, one of the differences between free and commercial offerings is the time for signatures to become available for detecting new threats as well as the quality of new initial signatures. For this illustration, we assume a commercial mail filtering product releases signatures for new threats one day after discovery and signatures block 99% of associated malicious messages without false positives.

For comparison purposes, we assume a free mail filtering product releases signatures for new threats three days after discovery and signatures block 95% of associated malicious traffic without false positives.

- Patch management–this intervention works by automating and/or simplifying the process of installing updates and patches to fix known vulnerabilities. This can increase the rate at which susceptible hosts are removed and directly transition to the recovered state without first becoming infected. Patches may be for vulnerabilities found in operating systems and/or installed applications. The complexity of a patch management solution may be very dependent on the homogeneity or heterogeneity of installed operating systems and applications in a large population. As previously noted, the computers or nodes are patched, not the associated users.

- User awareness training–this intervention aims to work by increasing user awareness to the possibility and signs of malicious messages to reduce the overall likelihood of users to open infected or malicious email messages. For illustration purposes, we assume an awareness training campaign decreases the overall user population expected likelihood value for opening infected messages by 4%.

Table 4.4 summarizes the intervention options and identifies some of the combinations of interventions explored. When applicable, differences from the baseline scenario parameters are indicated. Also included in the table is an estimated cost value for an intervention or combination of interventions. The options shown

Table 4.4: Intervention properties.

Index	Description	Cost ($)	Block delay	Block rate	Patch rate	User likelihood
1	Baseline	0	-	0	-	-
2	Free filter (FF)	10k	3	0.95	-	-
3	Pay filter (PF)	55k	1	0.99	-	-
4	Patch mgt (PM)	30k	-	0	+0.025	-
5	User training (UT)	25k	-	0	-	–0.04
6	FF & PM	40k	3	0.95	+0.025	-
7	FF & UT	35k	3	0.95	-	–0.04
8	PM & UT	55k	-	0	+0.025	–0.04

satisfy a cost constraint that intervention implementations must not exceed $60,000 ($60k). This constraint was imposed as an example of a constraint due to finite resources. Cost aspects are further discussed in the next section.

4.6.3 Costs related to outcomes and interventions

Two cost aspects considered when comparing interventions are the cost of the intervention itself as well as the costs related to the outcomes. The outcome costs include estimated costs for recovering or restoring infected hosts (per host), estimated costs for patching susceptible hosts (per host), and an estimated value related to increased risk of other types of malicious activity from infected hosts during the time they are infected. Table 4.4 shows values for each intervention option explored and Table 4.5 shows recovery and patching costs (per host) as well as the maximum value attributable to the risk of infected hosts being used for other types of malicious

Table 4.5: Outcome related costs.

Description	Cost ($)
Patched node (includes risk of bad patch)	1
Infected → Recovered node	5
Max. exposure (total time infected) risk cost	10,000

activity.

The associated risk cost is calculated using a logistic (or sigmoid) function based on the simulation results of the baseline case. The total time infected across hosts in a simulation is divided by the maximum total time infected across all hosts using the baseline scenario (which was 34,159 "host-days"; the theoretical maximum is 10,000 nodes * 90 days = 900,000 "host-days"), then an associated risk cost value is determined. The curve is centered around 17,500, which means if a simulation results in 17,500 total time hosts are infected, the associated risk cost value is $5,000 (or 1/2 of the maximum $10,000 cost). The cost value for patching hosts is intended to include both the time and effort to test and apply patches, but also additional costs that may be incurred if a patch does not work as intended and causes downtime. The cost value for recovering nodes is intended to include the time and effort to restore original functionality to a host while also ensuring it is no longer susceptible. These cost values are likely to be organization dependent and the values shown are for illustrative purposes only.

Figure 4.23 shows the implementation costs for each intervention (labeled as "Base costs") as well as box plots for range of patching, recovery, and risk costs for

1,000 simulations for each intervention option. Figure 4.24 shows box plots for all of the outcome related costs (patching, recovery, and risk) combined while Figure 4.25 also adds in the implementation costs for each option to show the range of total costs. While total costs may be one of the factors considered when comparing intervention options, reduction of uncertainty may be another factor used during evaluation of choices.

Looking at the total costs shown in Figure 4.25, intervention option 2 (free filtering) has the lowest median cost based on simulations (except for the baseline scenario which involves no interventions). Intervention option 7 (free filtering combined with user awareness training) is in the middle in terms of median cost of interventions based on the simulations, but has least variability among the interventions explored.

In this chapter, we presented a model for email virus propagation and discussed some potential influencing factors such as different network connection topologies and aspects of user behavior–most notably, the distribution of user likelihoods to open an infected email message. We show how decreasing connectivity levels decreases the total number of infected nodes and that this decrease is more variable for randomly connected networks than for scale-free networks with the same number of total connection links (for both uniform and exponential based user likelihoods for opening infected messages). We also covered how two available types of interventions (patching and blocking) can be explored with the model. This included an illustration of how intervention outcomes could be affected by network topology, level of connection links present, and different distributions of user likelihood values. The blocking intervention

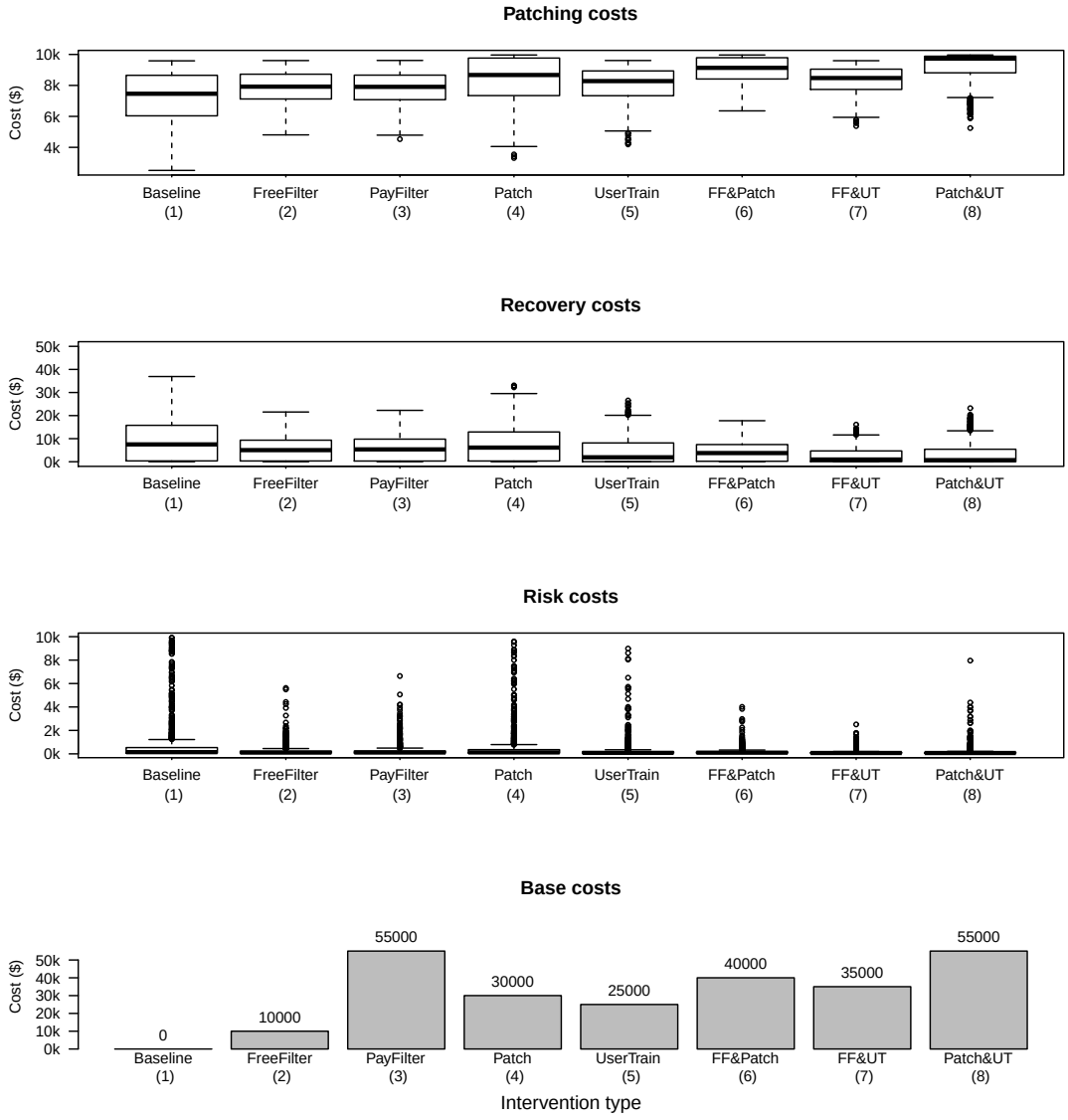

Figure 4.23: Intervention cost components.

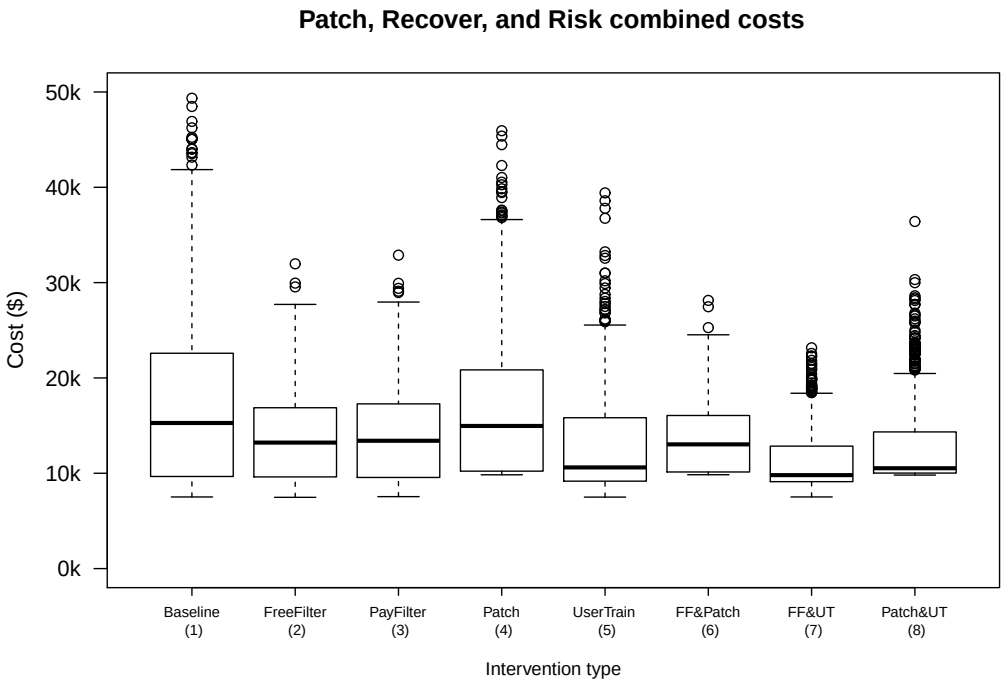

Figure 4.24: Intervention options outcome costs combined.

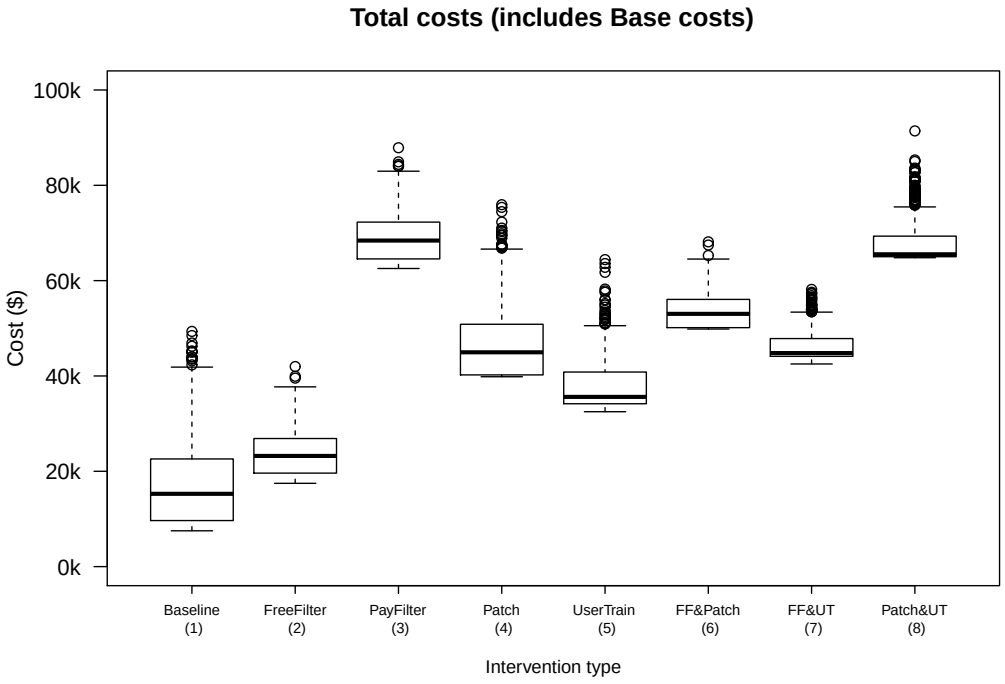

Figure 4.25: Intervention options total combined costs (implementation and outcomes).

appeared to become less effective for both connection topologies as the level of connection links decreases while the patching intervention appeared to retain its effectiveness better in the same circumstances. We provided an example illustrating the use of the models (in combination with cost information) to evaluate and compare outcomes resulting from different intervention implementations (including an intervention targeting user-awareness). This example demonstrated a method for using the models as a resource planning tool. The example showed how the different types of cost information could be combined with the model outcomes to provide final costs estimates which take into account intervention and outcomes costs. Several types of intervention implementations were modeled and compared. For the values used in the illustration, we saw that intervention implementations employing an aspect of user-awareness training were more cost effective than implementation involving patching.

Chapter 5: Software Reliability Growth and Time Series Models

5.1 Overview

When the symptoms or signs of a new type or strain of disease begins to affect human populations at noticeable and recorded levels, the underlying cause(s) may take time to determine. At this stage, often the frequency of symptom or disease occurrence is the only information available to make projections regarding future trends. The same can be true with regards to computer security incidents. The models presented in this chapter differ from the previously covered SIR and email propagation models. Two types of models described in this chapter can be used with historical data of incident occurrence over time to provide forecasts of future incident levels in time. These two types of models depend on different assumptions regarding the data being modeled. In Chapter 6, both types of models are applied to forecasting different types of computer security incidents to see which type of model may work best for certain types of incidents and to see if there are some general observations that can be made about forecasting the frequency of occurrence of computer security incidents.

The models presented in this chapter are less explanatory in nature and do not try to mimic or replicate the underlying dynamics and instead only try to describe

and forecast the observed data based on certain assumptions and simplifications regarding the underlying dynamics. While the causes of some types of computer security incidents may be known, such as when a particular computer worm is known to exploit a specific operating system vulnerability, the number of computers that contain this specific vulnerability and are exploitable is often unknown (the presence of the vulnerability itself may not be a sufficient condition for an event to occur; or some form of host-based antivirus software may provide additional protection). The causes of other types of incidents may never be determined and could be attributable to several different possible security failings. In some cases, these incident types are only detected based on how the computers are being used following an attack (such as acting as a relay for spam) and not by how the initial exploit occurred or was created. However, occurrences of these incident types may follow a pattern related to motivations behind an attack rather than with the distribution of vulnerabilities present in a population of computers.

5.2 Software Reliability Growth Models

In the context of software reliability, software reliability growth models have been used to describe the software defect detection process. In general, the assumptions driving software reliability include that software failures are caused by unpredictable events that do not have a corresponding remedy within the software and the failures that occur are independent. If the exploitation of vulnerable computers is viewed as being similar to the detection of faults in software, then software

reliability growth models may be suitable for forecasting the level of future incident occurrence. In this view, attackers take the role of being testers of the software and identifying software faults present in a system. As illustrated in the previous discussion regarding SIR models, if only a part of a larger population is being examined, the assumption that failures or incidents are independent may still be valid even for incidents that can spread from computer to computer. Some software reliability growth modeling concepts are applied to computer security incidents to see if this view is useful. Computers may be vulnerable due to flaws in software coding, but may also be vulnerable due to other factors such as improper or weak configurations.

5.2.1 Trend analysis

Trend analysis is a useful tool for gauging the applicability of using software reliability growth models. One test of reliability growth for a given time interval is to compare the expected number of failures in a given subinterval of time to the expected number of failures in a same size subinterval of time occurring earlier in the overall set. If, on average, the expected number of failures during the later subinterval of time is less than the expected number of failures for the earlier subinterval, the data indicate reliability growth [72]. The Laplace test is an analytical trend test that can be used to gauge reliability growth.

The time interval for a category of incidents is defined as being from the first day an incident of that type was observed through the last day such an incident was observed. Alternate starting points for time intervals were also considered,

such as vulnerability discovery dates published by CERT or software vendors (such as Microsoft). Release dates for various operating systems are another possibility. However, since some incidents are not easily connected to a single vulnerability and others work across multiple operating system versions and types, these methods were not used. It is also unknown precisely when detection capabilities for a particular incident type were added to the IDS or how much time occurred between watching for and observation of the first detected incident for a category of incidents. Therefore, the first day an incident was observed was selected to be the start of the time interval used.

Usually failure data is reported either as the time between failures (interfailure times) or as the number of failures for equally sized intervals of time. Because there are days in which a high number of incidents are detected and little specific information regarding the time of day the incidents occurred, the equation used for the calculating the Laplace factor is given by [73]:

$$u(k) = \frac{\sum_{i=1}^{k}(i-1)n(i) - \frac{(k-1)}{2}\sum_{i=1}^{k} n(i)}{\sqrt{\frac{k^2-1}{12}}\sum_{i=1}^{k} n(i)}, \tag{5.1}$$

where a time interval $[0, T]$ is divided into k equal units of time (days are used as the unit of time for our data) and $n(i)$ is the number of failures (incidents) observed during the i^{th} unit of time. According to reference [72], the practical interpretation of the Laplace test regarding reliability growth can be summarized as:

1. Negative values of the Laplace factor indicate a decreasing failure intensity

(reliability growth);

2. Positive values suggest an increasing failure intensity (reliability decay);

3. Values varying between -2 and +2 indicate stable reliability.

Plotting the Laplace factor over the time period of collected data is useful for viewing the trend evolution. Such a plot may indicate problems to be examined more closely by someone familiar with the context of the data. Changes in the slope of the plotted values may indicate a local shift in the reliability growth or decay of the data. This may merit closer examination to look for causal factors for the shift.

The information shown in a plot of the Laplace factor values can be used to aid the selection of reliability growth models that will provide better estimates. Models can be selected whose assumptions are more consistent with the data as indicated by the trends [74].

5.2.2 Software reliability growth models

Software Reliability Growth Models (SRGMs) are often used to describe and predict failures and faults in software systems. In many systems, reliability is assumed to increase over time as faults are found and fixed and as users become more familiar with the features of a system and can avoid or adjust to different failure states.

Most Non-Homogeneous Poisson Process (NHPP) models assume the cumulative number of failures that occur over a time interval can be described by a Poisson process with a mean value function $\mu(t)$ and the number of detected failures is proportional to the total number of faults in a system. The mean value functions

for NHPP models examined are nonlinear and defined by two parameters. Detected failures or observed computer security incidents may correspond to the $R(t)$ output of an SIR model, which may be represented using NHPP models. There are widely accepted and well-defined methods for estimating NHPP model parameters from observed data.

The results provided in Chapter 6 are based on four different NHPP models: the Goel-Okumoto (G-O) model [23], the S-Shaped model [24], the K-Stage Curve model [25], and the Duane model [26]. These models were selected because they represent some of the common NHPP models in use today. Three of them belong to the finite failures model class, except for the Duane model, which is an infinite failures model. In the NHPP models, the mean value function equals zero at time zero (zero failures found at the very beginning). The parameters of the model are estimated using the maximum likelihood estimation (MLE) technique. The likelihood function, $L(\theta|t_i)$, for the NHPP models is [75]:

$$L(\theta|t_i) = e^{-\mu(T)} \prod_{1 \leq i \leq N} \lambda(t_i), \tag{5.2}$$

where θ is a set of model parameters, T is the observation duration, and t_i are the observed failure times, $i = 1, 2 \ldots N$. The estimated model parameters are those which maximize the likelihood function. The natural logarithm of this likelihood, or the log-likelihood ℓ, is more convenient to work with when computing each estimate.

The mean value function for the G-O model is given by

$$\mu(t) = \alpha(1 - e^{-\beta t}). \tag{5.3}$$

The mean value function for the S-Shaped model is given by

$$\mu(t) - \alpha \left[1 - (1 + \beta t)e^{-\beta t}\right]. \tag{5.4}$$

The mean value function for the K-Stage model when $k = 3$ is given by

$$\mu(t) = \alpha \left[1 - e^{-\beta t}\left(1 + \beta t + \frac{(\beta t)^2}{2}\right)\right]. \tag{5.5}$$

The mean value function for the Duane model is given by

$$\mu(t) = \alpha t^{\beta}. \tag{5.6}$$

For three of the models (G-O, S-Shaped, K-Stage) the parameter α can be interpreted as the total number of errors in the system as $t \to \infty$, and β can be interpreted as the error detection rate. These interpretations do not apply to the Duane model.

5.3 Time Series Models

In the previous sections, the use of software reliability growth models was described for fitting and forecasting the number of computer security incidents occurring in some time interval. These types of models were chosen based on the

idea that computer security incidents could be viewed as analogous to faults in a system where external attackers are functioning as testers of the system.

However, other types of models with fewer assumptions than either SIR or SRG models can be applied. Since many computer security incidents may be the result of a computer virus or worm, techniques used in the field of epidemiology and infectious disease surveillance can also provide a useful framework. Chapters 3 and 4 examined aspects of the standard SIR model and an adaptation to include propagation by email. Previous discussion of the email propagation model illustrated how different interconnection topologies can produce varying epidemic patterns even when other factors are the same [59], and new strains (or variants) of viruses can emerge rendering existing vaccinations less effective. Models attempting to incorporate or account for the influence of these factors can increase in computational complexity. When it comes to modeling computer security incidents, the dynamics behind some incident types may not be well understood or simply may not be aptly described by or result from a contagious or infectious process model.

Another approach is to use time series analysis and modeling. Just as epidemic models based on the spread of diseases in human populations have been applied to modeling computer security incidents, techniques for applying time series models to diseases in human populations can also be applied to modeling computer security incidents.

5.3.1 Data transformations

Univariate time series models can be used to predict future values of a variable based only on its past measurements. These models do not rely on being able to measure or explain the causal factors underlying the behavior of the observed variable. Instead, the time series models are constructed to account for patterns in past movements in a manner useful for forecasting future behavior [76]. This is different from many other types of models.

While there are many types of time series models, we focus on a widely used and commonly applied class of linear models presented by Box and Jenkins [77] for univariate time series. These models combine autoregressive (AR) and moving average (MA) models into the combined autoregressive moving average (ARMA) models. In particular, we explore using autoregressive integrated moving average (ARIMA) models, which are one of the most commonly used models for univariate time series. The ARIMA model is a generalization of the ARMA model but includes a parameter for a differencing transformation of the data.

An AR model of order p denoted $AR(p)$ expresses the observed value at time t as:

$$X_t = \varepsilon_t + c + \sum_{i=1}^{p} \phi_i X_{t-p}, \tag{5.7}$$

where X is a weighted average of the previous p values in time with weights ϕ_i, ε_t an additional random error term and c a constant. The error term is generally assumed to be sampled from a white noise process of a mean of zero and constant variance.

Similarly, an MA model of order q denoted $MA(q)$ expresses the observed value at time t as:

$$X_t = \varepsilon_t + c + \sum_{i=1}^{q} \theta_i \varepsilon_{t-q}, \tag{5.8}$$

where X_t is a weighted average of the previous q random error terms with weights θ_i, ε_t an additional random error term and c a constant.

The $AR(p)$ and $MA(q)$ models can be combined to form an $ARMA(p, q)$ model expressed as:

$$X_t = \varepsilon_t + c + \sum_{i=1}^{p} \phi_1 X_{t-p} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-q}. \tag{5.9}$$

These models assume the underlying stochastic process being observed is time invariant or stationary. It is common to apply transformations to nonstationary data to obtain a stationary series before modeling.

Differencing is one technique for stabilizing the mean and it refers to successive applications of the following transformation:

$$\nabla X_t = X_t - X_{t-1}. \tag{5.10}$$

A backshift operator is commonly defined as:

$$BX_t = X_{t-1}, \tag{5.11}$$

which allows differencing to be expressed as:

$$\nabla^d X_t = (1 - B)^d X_t \tag{5.12}$$

with d as the number of repeated applications of the differencing transformation. An $ARIMA(p, d, q)$ model is an $ARMA(p, q)$ model but with the differencing transformation applied d times. For data that exhibits cycles or seasonality, a form of the backshift operator can also be applied to remove these effects and can be incorporated into a multiplicative seasonal ARIMA model (SARIMA), see [78] for more details.

The use of different transformations can be explored if the variance changes over time. Common transforms include logarithm, square root, and reciprocal. Mean-range plots may indicate if a particular transformation is appropriate.

It is also common to plot and examine the sample autocorrelation function (ACF) to check for seasonality in the data and to assist with selection of model parameters. The sample ACF provides an estimate of the interdependency between data points X_t and X_{t+k} usually expressed as a function of k where k is referred as the lag value. Stationary data will have ACF values that decay towards zero quickly as k increases.

Another related function is the partial autocorrelation function (PACF). Patterns in the ACF and PACF plots can sometimes indicate order values to try for the MA and AR parts of the models, respectively. For a discussion of ACF and PACF properties for different $MA(q)$ and $AR(p)$ ordered processes, see [79].

5.3.2 Model selection criteria

Because time series models involve fewer assumptions and the model order (p, q) is usually not known in advance, it is common to generate many different time series models and then select certain ones to retain and apply. This is different from the maximum likelihood approach to parameter estimation used by software reliability growth models. Two information-based selection criteria are typically used for choosing model order (p, q) when applying to actual incident data. Examining the residuals (also referred to as innovations) of one-step-ahead model predictions can provide an indication of model fit. However, to ensure the best forecasts, it is important to avoid selecting model orders (p,q) which may overfit the data and forecast poorly.

Akaike [80] described an approach to selecting models that attempts to minimize a measure consisting of two parts. The first part is a term to measure model fit and the second part is a penalty term for adding parameters (since including more parameters might result in overfitting). The form presented in [81] is used and identified as AICc. An equation for calculating AICc is given by:

$$AICc = ln\left(\frac{RSS}{n}\right) + \frac{(n+k)}{(n-k-2)}, \tag{5.13}$$

where RSS is the sum of squared residuals, n is the number of observations, and k is number of parameters $(k = p + q)$.

A second information criterion examined uses a different penalty term based

on Bayes factors [82] and is known as Bayesian information criterion (BIC), but can also be referred to as Schwarz information criterion (SIC). The equation for calculating BIC is given by:

$$BIC = ln\left(\frac{RSS}{n}\right) + \left(\frac{k \cdot ln(n)}{n}\right), \tag{5.14}$$

where RSS is the sum of squared residuals, n is the number of observations, and k is number of parameters ($k = p + q$). Chapter 6 provides examples of using AIC and BIC for selecting model order for time series models applied to computer security incident data.

This chapter reviews two types of models that can be applied to computer security incident data for forecasting purposes. These include several software reliability growth models applicable for modeling non-homogeneous Poisson processes. As described in Chapters 3 and 4, observed computer security incident data may correspond to the $R(t)$ output of an SIR based model for some incident types and modeled as an NHPP. Software reliability engineering practices provide widely used and accepted methods for estimating NHPP based SRGM parameters from observed data. ARIMA time series models are also presented as an additional type of model (requiring fewer assumptions) with forecasting applications. The process for estimating model parameters from data and the available criteria for selecting models are also discussed. Examples of applying these models to computer security incident data are presented in Chapter 6.

Chapter 6: Illustrations using Campus Data

Previous chapters described and illustrated several different types of models and aspects applicable to computer security incident data. This chapter shows some of the ways these models can be applied using actual incident data. Some aspects of each type of model previously described will be applied to and illustrated with collected incident data from a university network.

6.1 Description of data

The Division of Information Technology (DIT) at the University of Maryland provided the data set used in this analysis. This data set consists of almost 12,000 security incidents recorded over a period of about 9 years (from June 2001 through July 2010). This data set includes 51 different incident types. The number of incidents detected in a single day varies from 0 to 580.

The incidents recorded were based on three sources:

1. an intrusion detection system (IDS) (Snort [83] using a combination of regular rules and in-house rules),

2. reports from users, and

3. reports from other system administrators.

Since recorded incidents led to the blocking of the suspected computer's IP address, DIT verified the authenticity of each incident. Therefore, all incidents obtained from these three sources were manually reviewed. DIT launched port scans and packet captures to validate the suspicious behavior of identified hosts. Based on the IDS rule that raised an alert, about 60% of the alerts were inconclusive (e.g., when a detection rule was too broad). Among the remaining 40%, about half led to the direct action of DIT blocking the IP address and half required a confirmation. Among the incident alerts, very few were reports from users. The reports from other system administrators were defined as incidents in roughly 75% of the cases based on the source trustworthiness.

Because of the method used by DIT to validate the incidents, all incidents were real. Thus, there are no false positives among the incidents reported. However, the number of undetected attacks and intrusions that did not lead to a security incident is not quantified.

In this chapter, we mostly focus on the ten incident types that occurred most frequently in the data set between June 2001 and March 2007 (many of the incidents occurred between 2002 and 2005). The ten types of incidents out of 51 total represented only ≈20% of the incident types classified. However, these ten incident types accounted for ≈76% of the number of incidents in the full dataset. Details on the ten incident types examined for this time period are provided in Table 6.1. Table 6.1 contains the number of incidents recorded and the incident type. Most

incident type names are self-explanatory. Note, however, that "nethicsreq" usually represents the identification of an illegal use of copyrighted material (such as music and movies). The "Start Date" indicates the date when the first incident associated with that incident type was recorded. The "End Date" represents the date when the last incident associated with the incident type was recorded or the end data of the data collection, whichever is earlier. "# of Days" is the inclusive number of days in the interval bounded by "Start Date" and "End Date". The "Transfer" column indicates if a particular incident type is likely to be transmissible. A value of "Direct" means it is possible for that incident type to spread directly from computer to computer without user interaction. A value of "Email" means it is likely for that incident type to spread through infected emails messages. A value of "Other" means an incident either is not likely to be transmissible or the method is unknown. The value of "Multiple" is used when considering all incident types. It is possible for incident types to actually include variants and the method of transmission may not be the same for all variants. The raw data for some incident types is provided in Appendix A.

Figure 6.1 shows a timeline of the intervals for all of the data and for the top ten most frequent incident types. Figure 6.2 shows the cumulative number of all the incident types for the full time interval. There is a noticeable sharp increase around day 790. This increase is attributable to the start of the "worm_blaster" incident detections. Similarly, around days 244 and 994 are two other examples of a rapid increase in the number of incidents in a short period of time. Day 244 coincides with the start of detections for "virus_klez" incidents, and there were a large number

Incidents	# of Incidents	Start Date	End Date	# of Days	Transfer
All data	11966	6/14/2001	3/14/2007	2100	Multiple
worm_msblast	2133	8/12/2003	9/12/2003	32	Direct
virus_generic_bot	1940	6/8/2004	3/12/2007	1008	Other
virus_klez	1118	2/12/2002	12/2/2003	659	Email
bagle_worm	849	1/20/2004	11/28/2005	679	Email
irc_bot	690	4/8/2002	3/14/2007	1802	Other
virus_agobot	589	10/13/2003	9/13/2005	702	Email
rogue_ftp	500	4/19/2002	2/7/2007	1756	Other
nethicsreq	509	11/12/2001	3/14/2007	1949	Other
spamrelay	429	6/11/2002	3/9/2007	1733	Other
worm_nachi	317	9/16/2003	4/1/2004	199	Direct

Table 6.1: Summary of incident data.

of "bagle_worm" incidents detected around day 994, which were likely due to a new variant of this worm being released at the time. Not all incident types included in the data set are easily linked with a particular cause and in some cases the incident type name represents a detected symptom or behavior (such as "spamrelay") instead of identifying an underlying cause.

6.2 Approximate Bayesian Computation

It is usually desirable to estimate or infer parameter settings for these models to closely approximate a particular setting or environment. Models can be used to evaluate interventions for a particular environment, however, measuring or identifying the model parameter values to use for a particular environment may not be simple. For example, the full set of email contacts and links within an organization may not be well known, or the distribution of user likelihoods to open a infected messages may also be difficult to measure. Approximate Bayesian Computation (ABC) [84] is one method for model parameter inference in cases where a model is complex or

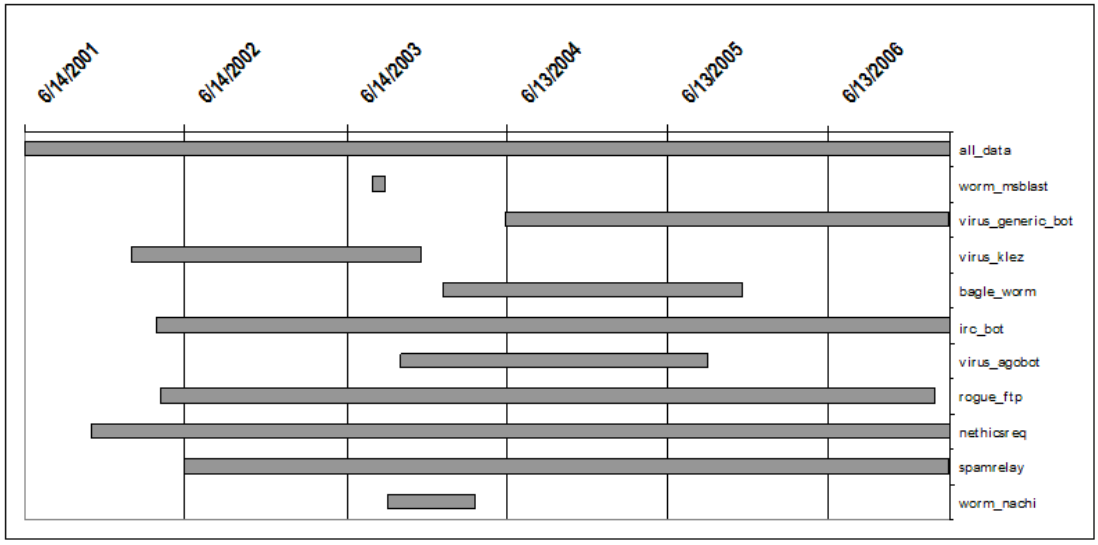

Figure 6.1: Timeline of incident data.

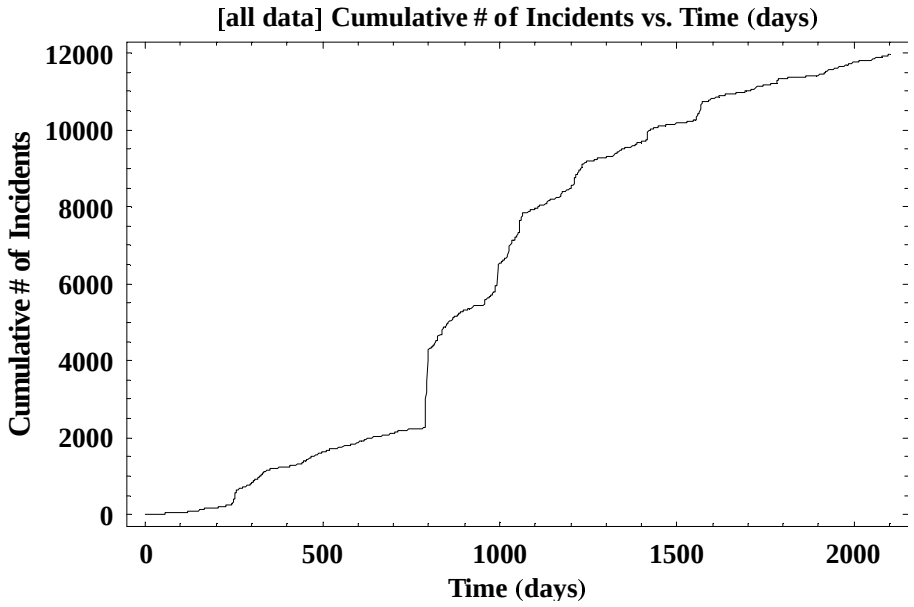

Figure 6.2: Cumulative number of incidents.

not well suited to other parameter inference methods (such as maximum likelihood estimation). ABC can also be used for selecting between several types of models. We use ABC for both purposes when exploring stochastic email propagation models using real incident data. We use MLE for software reliability growth model parameter inference because it is less computationally expensive, and AICc/BIC for time series model selection because they include a penalty term based on the number of model parameters to reduce the chances of overfitting the data.

Essentially, ABC involves generating model outcomes for a range of parameter values and comparing these outcomes with a set of observations. A distance value or some measure of fit between the generated outcomes and the set of observations is calculated. Then a threshold is used to retain a smaller subset of the specific models that generated the outcomes. Several best fitting models are retained to help avoid overfitting. The threshold could be in the form of some maximum allowable distance value between model outcomes and observations or it could be simply a number or percentage of model outcomes with the smallest distance values from the set of observations. The parameters settings of the specific models retained in the selected subset can then form an estimate of a posterior distribution of the parameter values.

In [85], ABC is compared with MCMC for the standard SIR model and similar posterior parameter distributions are obtained. An ABC based method for selecting between different SIR based models is presented in [86].

Although ABC can be used parameter value inference, we will first explore using it for identifying or classifying which model types may fit better with the underlying processes that produced a set of observations. In this case, we generate

model outcomes over a range of parameters for several model types. Then we combine the model outcomes and retain a certain number of the models generating outcomes with the closest fit values to the set of observations. We use the sum of the Euclidean distances of between model outcome data points and test case data points as our measure of fit value. From this set of closest fitting models, we then count how many resulted from each type of model and use this as an indicator of the relative likelihood the set of observations resulted from a specific model type for the range of parameters used.

6.3 SIR Models

For this section, we explore two types of SIR models and generate a test case of observations for each. Model outcomes are then computed over a range of parameter values to compare outcomes of both model types to both test cases. We do this for different ranges of parameter values to explore how the range of parameter values used may affect model identification. We then compare the two model types to two sets of incident observations to identify which models or model features are most applicable to the incident observations. This is done by selecting a range of parameter values and generating model outcomes for comparison to the incident observations.

6.3.1 Test cases

Previously, we presented examples of how patching and the rate of patching in a network of hosts or nodes could affect the spread of a worm or virus that propagates through direct (node-to-node) communication between computers. We presented two model types to illustrate two ways patching may occur (either at a constant rate or at a rate proportional the number of remaining unpatched nodes). In this section, we generate a set of test case observations for each model type. We define the two model types as follows:

- Model A–uses a constant patching rate parameter where the chances of particular susceptible node being patched are the same at each time step.
- Model B–uses a variable or proportional patching rate parameter where the chances of particular susceptible node being patched are proportional to the number of unpatched or susceptible nodes at each time step (so as the number of susceptible nodes decreases in time, their chances of being patched also decreases in time).

When applicable, the same parameter values are used for generating the set of observations for the test cases. All test cases and model outcomes are based on a fully connected network topology.

6.3.2 Test parameter sets

We explore how using different parameter value ranges for generating model outcomes are to see how this may affect identification of the original model type used to generate at test case. It is expected the distribution of distance values for the generated model outcomes will flatten or diffuse as the range of parameter values broadens to include values farther away from the original parameter values used to generate the test cases. However, it is unknown if or how much effect this may have on the closest fitting model outcomes that will be retained for identifying or classifying the model type originally used to generate the test case. Four parameter sets are used and are described as follows:

- Parameter Set 1–this set is same as the parameters used for generating the test cases. There are 5,000 nodes which are fully connected; $\beta = 1.0$; $\gamma = 0.5$; $\nu = 0.025$.

- Parameter Set 2–same as Parameter Set 1, except allow value for β to range uniformly between [0.9, 1.1]; the value for γ to range uniformly between [0.4, 0.6]; the value for ν to range uniformly between [0.015, 0.035].

- Parameter Set 3–same as Parameter Set 2, except expand the range for β to [0.7, 1.3]; value for γ in the range [0.2, 0.8]; and the value for ν to range uniformly between [0.005, 0.045].

- Parameter Set 4–same as Parameter Set 3, except for the addition of allowing the total number of nodes to vary uniformly within the range [4750, 5250].

6.3.3 Test results and observations

For each model type (A and B) 2,500 simulations are run for each of the four parameter sets described above. All resulting model outputs are compared with each test case to obtain a distance or fit value. For a particular test case, outputs for both model types (using the same set of parameter values) are combined (so there are 5,000 total model outputs being compared for each test case for each parameter set). The models that produced the 50 closest set of outputs to the test case are retained and then the number of each model type in this set of 50 is counted and used as an indicator of how likely it is that the test case data was originally generated by a model of this type.

Figure 6.3(a) shows a time curve of outcomes that make up Test Case A (originally generated by Model Type A and parameter values stated earlier). Figure 6.3(b) shows a histogram of all of the model fit values produced by Model Type A using Parameter Set 1 when compared with Test Case A. Similarly, Figures 6.3(c)-6.3(e) show the same but using Parameter Sets 2-4 respectively. The figures show that the overall distribution of model fit values flattens out as the set of parameter values expands (except for the spike around 5,000 which corresponds to cases where the infection does not spread and the final total number of infected cases is very low). However, in the range of parameter values used for these simulations, there are several model output results which are close in distance to the test case.

Figure 6.4 shows the number of each model type that created the closest 50 model outputs to the original test case for each of the parameter ranges explored.

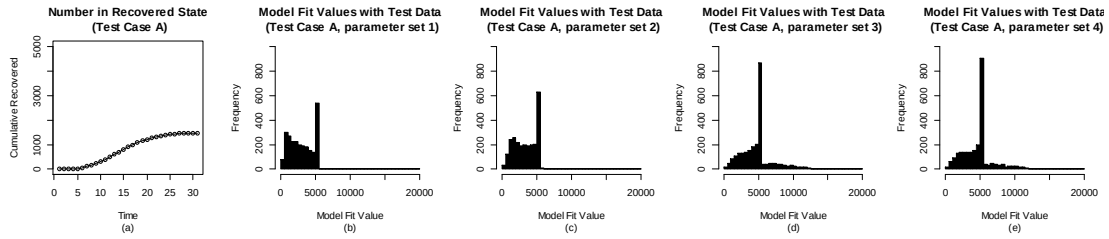

Figure 6.3: Test Case A and histograms of model fit values for Model A for different parameter sets.

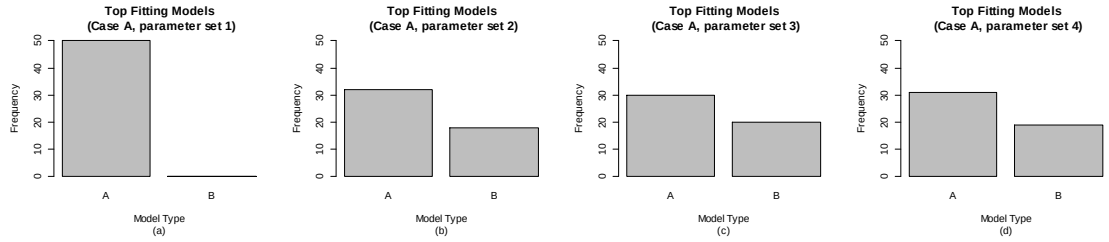

Figure 6.4: Distribution of model types for the 50 best fitting model outputs for Test Case A for different parameter sets.

Figure 6.4(a) shows all of the closest 50 model outputs to Test Case A were generated by Model Type A. Figures 6.4(b)-6.4(d) show that about 30 of the closest 50 model outputs to Test Case A were generated by Model Type A, while about 20 were generated by Model Type B (for Parameter Sets 2-4). Based on these results, one could infer that there is a 60% chance that Test Case A resulted from Model Type A and a 40% chance Test Case A resulted from Model Type B (when considering the parameter ranges in Parameter Sets 2-4).

6.3.4 Application using incident data

In this section, ABC is used to compare actual incident data to the two types of models described in the previous section. The goal is to see if the method can provide some indication of which model type that would better describe the dynamics

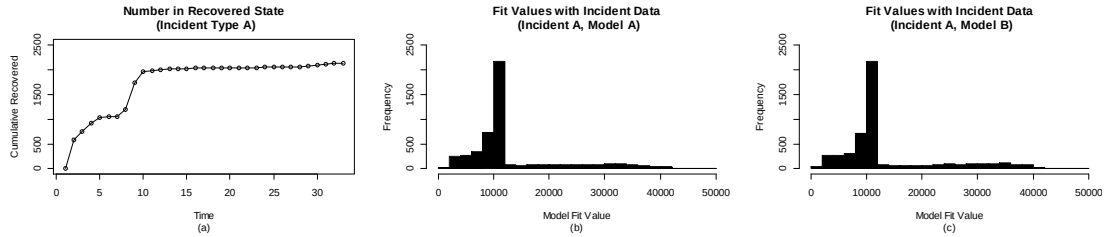

Figure 6.5: Incident Type A and histograms of model fit values for model types A and B.

of an environment from a set of models and a range of parameter values.

6.3.4.1 Incident data

From the original incident data set, occurrences of two different types of incidents that could potentially spread directly from computer to computer were selected. Although referred to in this section as Incident Types A and B, they correspond to the "worm_msblast" and "worm_nachi" incidents, respectively, presented earlier in Table 6.1. The full time period for "worm_msblast" is used, but the time interval for "worm_nachi" is limited to the first 120 days (when the bulk of the incidents occur). Figure 6.5(a) shows the plot of values for Incident Type A.

6.3.4.2 Applied parameter ranges

For each of the two types of models described in the previous section, 5,000 model outcomes are computed using a range of model parameter values, for a total of 10,000 model outcomes each for Incident Types A and B. Each of the model outcomes is compared with each type of the selected incident data and a distance or fit value is calculated. The following parameters and value ranges are used:

- β (infection rate) using a uniform random distribution with the range of [0.1, 5.0].
- γ (removal rate) using a uniform random distribution with the range of [0.1, 5.0].
- ν (patching rate) using a uniform random distribution with the range of [0, 0.1].
- The total number of nodes was kept fixed with a value of 10,000.

6.3.4.3 Applied results and observations

Figure 6.5(a) shows a plot of the data for Incident Type A. Figures 6.5(b) and 6.5(c) show histograms of the model fit values for Incident Type A to model outcomes generated by model types A and B, respectively, over the parameter value ranges described above. While the histograms of model fit values appear to be very similar, the important visible difference is that the frequency of Model B outcomes at the lowest model fit values is slightly higher than the frequency of Model A outcomes for the same model fit values. This is the region of fit values for the retained models.

After all model outcomes and fit values were calculated, the distribution of model types that produced the 20 best fitting model outcomes for each email incident type were examined. These results are shown in Figures 6.6(a) and 6.6(b) for incident types A and B respectively. Figures 6.6(c) and 6.6(d) show histograms of the model fit values for the 20 retained best fits for each incident type (from both Model A and Model B types).

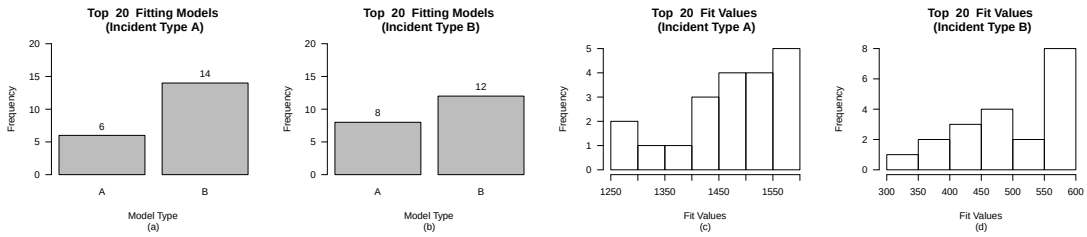

Figure 6.6: Frequency of model types generating best 20 fitting model outputs and histograms of best 20 fit values.

We make the following observations:

- From Figure 6.6(a), we could conclude that there is a 30% chance (6/20) the observations for Incident Type A resulted from Model Type A (constant vaccination rate) and a 70% chance (14/20) the observations resulted from Model Type B (variable vaccination rate).

- From Figure 6.6(b), we could conclude that there is a 40% chance (8/20) the observations for Incident Type A resulted from Model Type A (constant vaccination rate) and a 60% chance (12/20) the observations resulted from Model Type B (variable vaccination rate).

- For both incident types, Model Type B (variable vaccination rate) is the most likely model type (of the two tested) to have generated the observed data.

6.4 Email Propagation Models

For this section, four types of models are explored and test cases for each are generated. The four types of models vary in terms of network connection topologies and distributions of user likelihood values. Model outcomes are generated to compare

all model types to all test cases. This is done for several ranges of parameter values to explore how the range of parameter values used may affect model classification. A range of parameter values is selected for generating model outcomes for comparing the four model types to three sets of incident observations to determine which models or model features are most applicable to the incident observations.

6.4.1 Test cases

Chapter 4 presents illustrations of how network connection topology and the distribution of user actions in the network can affect the spread of an email virus. Four model types which cover the ways these properties can be combined are explored further and a set of test case observations for each model type is generated. The four model types used are described as follows:

- Model A–uses a randomly connected network topology and a uniform random distribution for assigning user likelihood values for opening infected messages.
- Model B–uses a randomly connected network topology and an exponential based distribution for assigning user likelihood values for opening infected messages.
- Model C–uses a scale-free network topology and a uniform random distribution for user likelihood values for opening infected messages.
- Model D–uses a scale-free network topology and an exponential based distribution for assigning user likelihood values for opening infected messages.

When applicable, the same parameter values were used for generating the set of observations for the test cases. Approximately the same total number of edges or links are used for the randomly connected and scale-free connected network topologies. The uniform random and exponential based distributions for assigning user likelihood values for opening infected messages have similar expected values for the population.

6.4.2 Test parameter sets

Different parameter ranges for generating model outcomes are used to explore how this may affect model classification. Four parameter sets are used and are described below:

- Parameter Set 1–this set is same as the parameters used for generating the test cases. There are 5,000 nodes with about 2,500,000 edges or links (20% connected); $\beta = 1.0$; $\gamma = 0.3$; an expected value of 10% for user likelihood for opening an infected message.

- Parameter Set 2–same as Parameter Set 1, except allow value for β to range uniformly between $[0.9, 1.1]$; the value for γ to range uniformly between $[0.2, 0.4]$; the number of links to range from 17% to 23% of a fully connected graph; and the expected user likelihood value in the interval $[0.07, 0.13]$.

- Parameter Set 3–same as Parameter Set 2, except expand the range for β to $[0.8, 1.2]$; value for γ in the range $[0.1, 0.5]$; number of links from 15% to 25% of a fully connected graph; and expected user likelihood values in the interval

$[0.05, 0.15]$.

- Parameter Set 4–same as Parameter Set 3, except for the addition of allowing the total number of nodes to vary uniformly within the range $[4750, 5250]$.

It was expected the distribution of distance values for generated model outcomes would flatten or diffuse as the range of parameter values broadened farther away from the parameter values used to generate the test cases. However, what was unknown was if or how much affect this might have on the closest fitting model outcomes used for identifying or classifying the model type originally used to generate a test case.

6.4.3 Test results and observations

For each model type (A, B, C, and D) we run 2,500 simulations for each set of parameter values described above. All resulting model outputs are compared with each test case to obtain a distance or fit value. For a particular test case, outputs for each model type (using the same set of parameter values) are combined (so there are 10,000 total model outputs being compared for each test case for each parameter set). The models that produced the 100 closest set of outputs to the test case are retained and then the number of each model type in this set of 100 is counted and used as an indicator of how likely it is that the test case data was originally generated by a model of this type.

Figure 6.7(a) shows the time curve for Test Case D (originally generated by Model Type D and parameter values stated earlier). Figure 6.7(b) shows a histogram of all of the model fit (or distance) values produced by Model Type D using Parameter

Set 1 when compared with Test Case D. Similarly, Figures 6.7(c)-6.7(e) show the same but for Parameter Sets 2-4 respectively. We observe that the overall distribution of model fit values flattens out as the set of parameter values expands. However, for the range of parameter values used in these simulations, we see there are still several model output results which are close in distance to the test case (i.e. have low model distance values).

Figure 6.8 shows the frequency of each model type making up the closest 100 model outputs to the original test case. We see in Figure 6.8(a) that 98 of the closest 100 model outputs to Test Case D were generated by Model Type D while 2 were generated by Model Type B. Based on these results, we would infer that there is a 98% chance that Test Case D resulted from Model Type D and a 2% chance Test Case D resulted from Model Type B (when restricting parameter values to those in Parameter Set 1).

In Figures 6.8(b)-6.8(d), we see that as the range of parameters used increases, it becomes more difficult to discriminate between Model Type D and Model Type B as the most likely model type responsible for creating the test data. However, we see that for the broadest range of parameter values used (Parameter Set 4), discriminating between models using uniform random user likelihood distributions (Models A and C) and models using exponential user likelihood distributions (Models B and D) is still possible in this example.

When comparing Model Types B and D, we can see from Figure 4.15(c) and Figure 4.16(c) that at similar network connectivity levels, the difference between the time curves of nodes in the recovered state is slight for the randomly connected

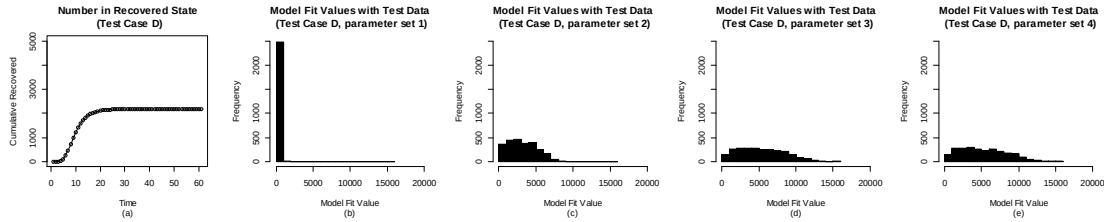

Figure 6.7: Test Case D and histograms of model fit values for Model D for different parameter sets.

and scale-free connected examples shown. This test case result is consistent with these previously shown examples. For the range of parameters used in the test cases, results show more differentiation due to the different types of user likelihood distributions than due to the different types of network topology.

Figure 6.9 shows the prior and posterior distributions for the user likelihood parameter for Model D with Test Case D for each parameter set. The prior distribution density (shown as a dashed red line) shows all of the initial 2,500 sampled parameter values used for each parameter set. The posterior distribution density (shown as a density histogram in grey) is obtained from the user likelihood values used by the Model D simulations present in the top 100 models retained. The number of each model type retained is indicated in Figure 6.8. We see that the posterior distribution is a closer reflection of the actual parameter value (0.10, shown as a vertical dotted line) used to generate Test Case D compared to the prior (or sampling) distribution. This is true even as the range of sampled parameter values increases.

6.4.4 Application using email incident data

In this section, we use ABC to compare actual email virus incident data to the four types of models described in the previous section. Our goal is to see if the

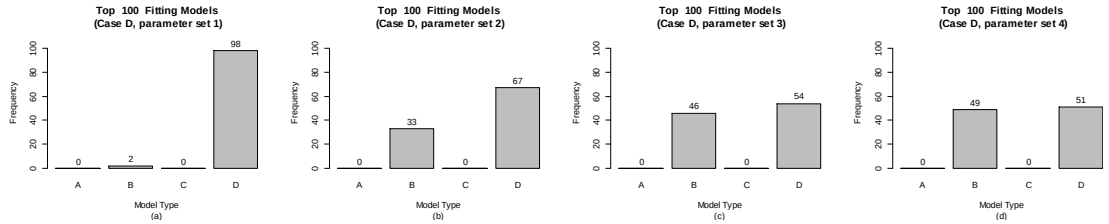

Figure 6.8: Distribution of model types for the 100 best fitting model outputs for Test Case D for different parameter sets.

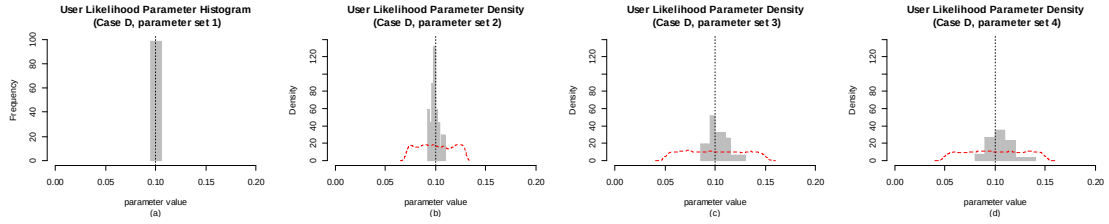

Figure 6.9: User likelihood parameter distributions *(prior in red, posterior in grey)* for Model Type D of the 100 best fitting model outputs for Test Case D across parameter sets. *[Actual User Likelihood Parameter value = 0.10.]*

method can provide some indication of the model type and parameter values that would better describe the setting from a set of models and a range of parameter values.

6.4.4.1 Email incident data

From the original UMD incident data set, we extracted occurrences of three different types of viruses that could potentially spread through email, we refer to them as Incident Types A, B, and C. Since some of the incident types potentially involved multiple variants which may also exploit different vulnerabilities, we selected a 120-day time window for each incident type to use for the analysis. The collected data reflects number of nodes removed/recovered over time (i.e. the $I \rightarrow R$ transition); no information is available regarding when an infection actually occurred (i.e. the

$S \rightarrow I$ transition). Although referred to in this section as Incident Types A, B, and C, they correspond to the "bagle_worm", "virus_klez", and "virus_agobot" incidents, respectively, presented earlier in Table 6.1. Figure 6.10(a) shows the plot of values for Incident Type C.

6.4.4.2 Parameter ranges

For each of the four types of models described in the previous section, we compute 6,250 model outcomes using a range of model parameter values, for a total of 25,000 model outcomes. Each of the model outcomes is compared with each type of the extracted email incident data and a distance or fit value is calculated. The following parameters and value ranges were used:

- β using a uniform random distribution with the range of $[0.1, 2.0]$.
- γ (removal rate) using a uniform random distribution with the range of $[0.01, 2.0]$.
- ν (patching rate) using a uniform random distribution with the range of $[0, 0.5]$.
- Number of links or edges as a percentage of a fully connected graph using a uniform random distribution with the range of $[2\%, 25\%]$.
- Expected value of user likelihood to open an infected message with the range of $[0.01, 0.4]$
- The total number of nodes was kept fixed with a value of 10,000.

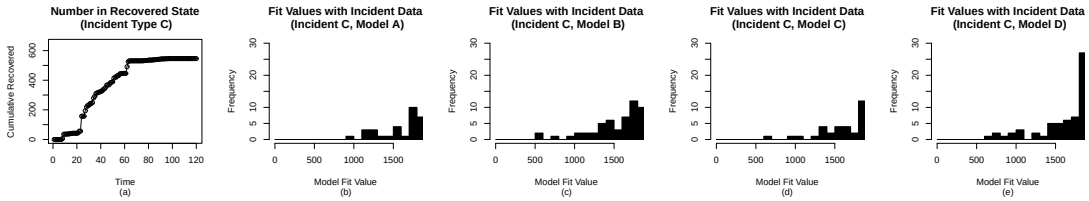

Figure 6.10: Type C and histograms of model fit values for different model types.

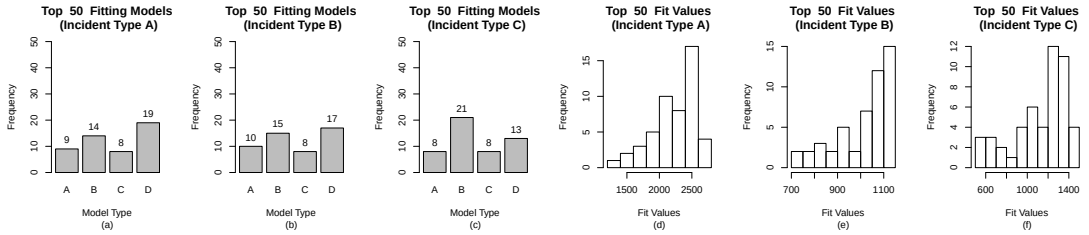

Figure 6.11: Frequency of model types generating best 50 fitting model outputs and histograms of best 50 fit values.

6.4.4.3 Results and observations

Figures 6.10(b)-(e) shows histograms of the model fit values for Incident Type C to model outcomes generated by model types A, B, C, and D, respectively, over the parameter value ranges stated above. The full range of model fit values is not shown, but includes all of the 50 best fitting models. After all model outcomes and fit values were calculated, we looked at the distribution of model types producing the 50 best fitting model outcomes for each email incident type. The distributions of retained models are shown in Figures 6.11(a)-(c) for incident types A, B, and C respectively. Figures 6.11(d)-(f) show histograms of the model fit values for the 50 best fits for each incident type.

Figure 6.12 shows prior and posterior distributions of parameters for the Model Type B simulations when compared to the Incident Type C data. The comparison

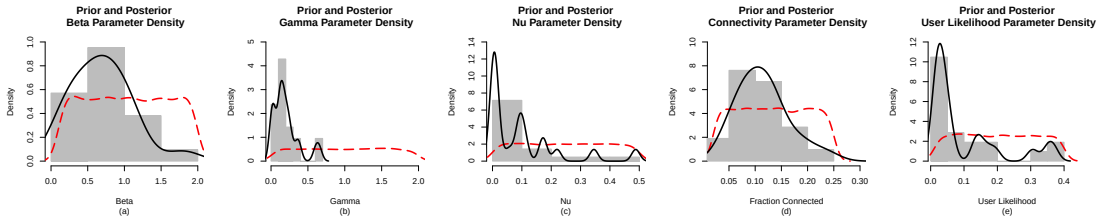

Figure 6.12: Incident Type C, Model Type B–parameter distributions *(prior density in red, posterior density in black, posterior density histogram in grey).*

of the prior and posterior distributions can be used to help gauge if an appropriate prior (sampled) range was used and/or if the number or percent of retained samples should be changed (which may also mean changing the overall number of simulations used). The obtained posterior distributions can be used to help guide future selection of prior distributions to sample from.

We make the following observations:

- From Figure 6.11(a), we could conclude that of the models tested, there is a 18% chance (9/50) the Incident Type A observations resulted from Model Type A, a 28% chance (14/50) from Model Type B, a 16% chance (8/50) from Model Type C, and a 38% chance (19/50) from Model Type D for the tested range of parameter values. Also, we could interpret the results to indicate there is 66% chance (33/50) the observations involved an exponential based distribution of user likelihoods to open infected email messages compared to a 34% chance (17/50) of a uniform random distribution of likelihoods. However, it is possible that a different distribution (other than uniform random or exponential) is actually responsible. The ABC method only provides estimates of aspects included in the testing process. Similar inferences can be stated based on

Figures 6.11(b) and 6.11(c).

- For two of the three incident types, Model Type B (randomly connected network topology and exponential based distribution of user likelihood values to open infected messages) is the most likely model type (of those tested) to have generated the observed data. The remaining incident type (C) was best matched to Model Type D (scale-free network topology and exponential distribution of user likelihood values to open infected messages).

- For the three incident types, the retained models are more likely (66%, 64%, and 68% for Incident Types A, B, and C, respectively) to have used an exponential distribution than a uniform random distribution for the user likelihood parameter.

- Figure 6.12(e) indicates better fitting model B simulations to Incident Type C data are more likely to have user distributions based on smaller (5% or less) average expected values of user likelihoods to open infected messages. This may indicate there are a small number of users who are more likely to open infected messages and implementing interventions that account for this could be beneficial (e.g. targeted blocking or patching of computers used by these users).

This section presented and tested a method for using Approximate Bayesian Computation to evaluate model types for a set of test case data. This method was applied to three email virus incident types. For the model types tested and for the range of parameter values explored, model types using an exponential based

distribution of user likelihood of opening infected messages produced the best fit values as well as models based on a scale-free network topology.

6.5 Software Reliability Growth and Time Series Models

The previous applications of SIR and Email Propagation models focused on using the models to compare which features of the models best describe or account for the observed incident data for the applicable incident types. However, not all computer security incident data involves processes which are transmissible or fit with an infectious disease model approach. In these cases, while it may be difficult to create an explanatory model for the anticipated impact of a particular intervention (such as implementing a policy change), it is still desirable to gauge the impact of intervention actions. One way to do this is to use data collected before an intervention is applied to project future levels and then compare these projections to actual incident levels after one or more intervention actions has be taken. Software reliability growth models and time series models are potential ways to make projections based on collected data.

6.5.1 Trend analysis and software reliability growth models

While incident data is not the same as software failure data in systems, there are some similarities. It is reasonable to assume for incidents that can be linked to a direct cause such as a particular worm or virus variant that exploits a single vulnerability, the frequency of detection will begin to decline at some point as infected

machines are removed and repaired. Vulnerabilities in remaining machines will also be patched against the exploit to prevent future incidents of the same type as user awareness increases and antivirus software is updated. User awareness of particular issues and incident types may also improve over time and lead to a decrease in observed incidents. These possibilities may result in observed incident data which show a variable and declining rate of incident occurrence for which software reliability growth models may provide a good fit and may be useful for making projections.

Figure 6.13 shows the Laplace trend values for the all incidents. The plot shows there are a few brief periods of rapidly increasing Laplace trend values followed by a slower decline in values. The first rapid increase is attributable to detections of "virus_klez" incidents. The largest increase in Laplace trend values occurs around day 790 and coincides with the start of "worm_msblaster" incidents. The smaller increase but noticeable peak in Laplace factor values results from a sudden spike in the number of "bagle_worm" incidents recorded (likely due to the release of a new variant of the worm). While these changes can be seen when viewing the cumulative number of incidents as shown in Figure 6.2, they are more noticeable in the Laplace trend values as shown in Figure 6.13.

Although the cumulative total number of incidents may not fit well to a reliability growth model (indicated by the Laplace trend plot shown in Figure 6.13) in part due to the dynamic nature of the observed system (the number of networked devices fluctuates over time; the relative proportion and versions of operating systems in the system evolves as new operating systems and devices come to market) and the data not being fully consistent with some of the assumptions made by such

models, it seems reasonable to expect that some of the different types of incidents can be described with reliability growth models. Incident types with narrow and well-defined causes or vulnerabilities (such as "worm_msblast") are more easily fixed and patched against. These types of incidents are more likely to show characteristics in common with reliability growth models. Other types of incidents may not be attributable to a single cause or vulnerability (such as "spamrelay"). Reliability may improve as systems are identified and fixed, but reliability may also decrease or remain stable if new sources of vulnerabilities are found to cause incidents of the same type. This suggests that relating incident types to their causes instead of their symptoms may be useful for modeling purposes, however, obtaining the necessary information for this is not always practical for some incident types.

We now take a closer look at some of the individual incident data. The trend values for individual incident types are calculated using different time windows for each incident type. The time windows are based on when incidents of a particular type where first and last observed. This information is provided in Table 6.1 under the columns labeled "Start Date" and "End Date." Figure 6.14 shows the evolution of the daily Laplace trend values for the "worm_msblast" data. The negative values indicate reliability growth throughout the interval. Most reliability growth models can be applied to this data. The beginning part of Figure 6.15 indicates a decrease in reliability as illustrated by the positive Laplace factor values. Since applied models would need to allow for increasing failure intensity, the G-O [23] model would not be expected to work well for the "spamrelay" data. Figure 6.16 indicates an initial decrease in reliability followed by reliability growth as illustrated by the change in

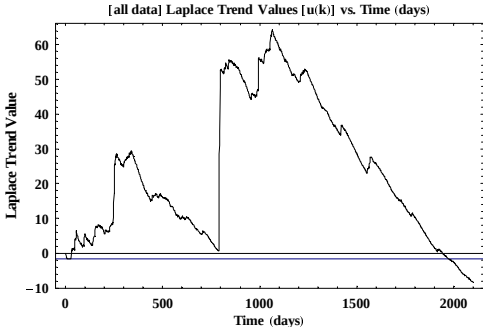

Figure 6.13: Laplace trend values for all incident data.

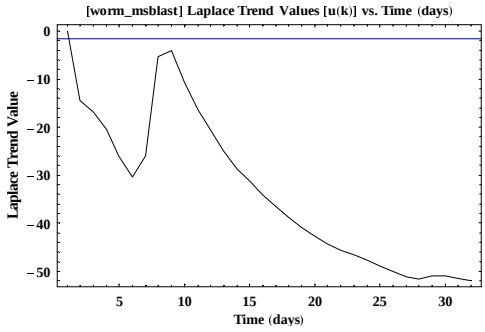

Figure 6.14: Laplace trend values for "ms_blast" incident data.

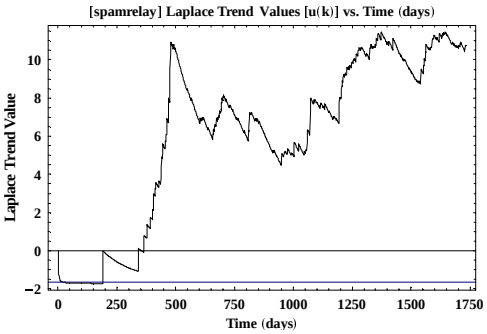

Figure 6.15: Laplace trend values for "spamrelay" incident data.

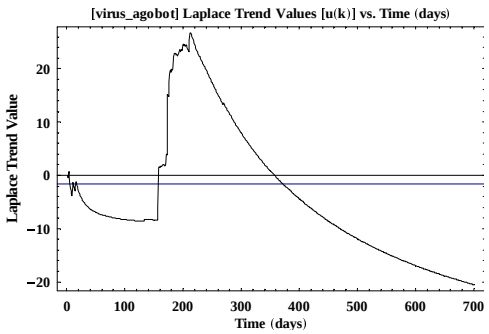

Figure 6.16: Laplace trend values for "virus_agobot" incident data.

Laplace factor values from positive to negative. An S-Shaped [24] or K-Stage [25] model might work best for the "virus_agobot" data. Laplace trend values were plotted each of the incident types considered in the paper, but not all are shown. Figures 6.14, 6.15 and 6.16 were included to illustrate examples of features to look for when considering models to apply. Some of the other incident types had more complex trend features. For example, "bagle_worm" has many small peaks across the considered time interval that likely coincide with the release of new variants of the worm.

Table 6.2 summarizes the chi-square model fit values for each model type for

Incidents	G-O Model Chi-squared	S-Shaped Model Chi-squared	K-Stage Model Chi-squared	Duane Model Chi-squared
All data	797,069	333,191	213,386	774,200
worm_msblast	496	3,049	10,804	3,807
virus_generic_bot	21,721	38,239	177,952	43,025
virus_klez	15,452	125,272	647,881	8,914
bagle_worm	10,975	15,115	25,594	52,068
irc_bot	25,486	251,787	7,966,766	17,292
virus_agobot	30,996	24,568	109,530	59,942
rogue_ftp	37,277	38,573	1,104,622	41,978
Nethicsreq	21,852	12,561	689,714	24,848
Spamrelay	57,250	13,724	2,997,065	10,662
worm_nachi	826	9,174	71,563	2,630

Table 6.2: Chi-square fit values for incident models over full time intervals.

the different incident classes. The chi-square goodness of fit is defined as:

$$\chi^2 \quad = \quad \sum_{i=1}^{T} \frac{(o_i - e_i)^2}{e_i} \tag{6.1}$$

where o_i is the observed cumulative number of incidents at day i and e_i is the estimated cumulative number of incidents at day i.

From Table 6.2, the G-O model ($\alpha = 2,141.1$, $\beta = 0.17428$) appears to have the best chi-square fit for the "worm_msblast" incident data and this is consistent with the Laplace trend evolution shown in Figure 6.14. The S-Shaped model ($\alpha = 591.34$, $\beta = 0.010963$) appears to fit the "virus_agobot" incident data better than the other models (also consistent with Laplace trend evolution shown in Figure 6.16), but the K-Stage model ($\alpha = 589.41$, $\beta = 0.016632$) provides the poorest chi-square fit. This is not expected as the initial increase in Laplace trend values indicates the G-O model will not be a good fit and could be expected to have a higher chi-square fit value than the S-Shaped and K-Stage models. Figure 6.15 shows mostly positive

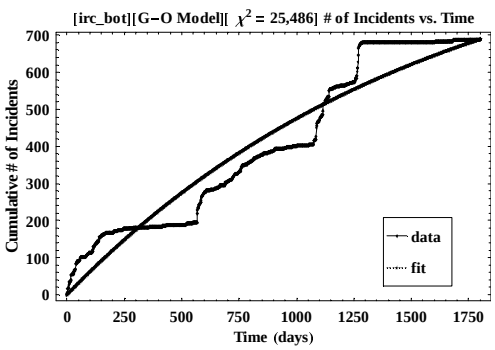

Figure 6.17: G-O model fit and data for "irc_bot".

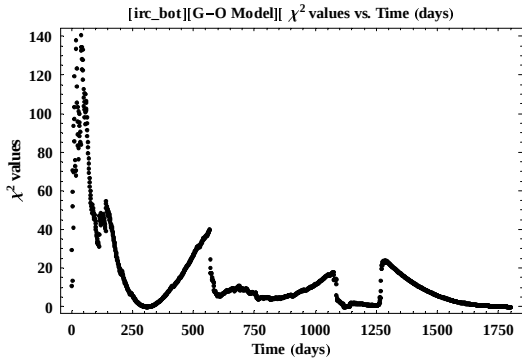

Figure 6.18: Chi-square residuals for G-O model fit of "irc_bot" data.

trend values and the Duane model ($\alpha = 2.8061 \times 10^{-4}$, $\beta = 1.9095$) provides the best chi-square fit for the "spamrelay" incident data.

For the "irc_bot" incident type, Figure 6.17 shows a G-O model fit to the data while Figure 6.19 shows the K-Stage model fit to the same data. Visually the models appear similar (except for the first few days). However, as shown in the Table 6.2, the calculated chi-square value for the K-Stage model is extremely high when compared to the calculated chi-square value for the G-O model. Comparing the plot of chi-square residual values (Figures 6.18 and 6.20) shows the influence of the first few predicted values on the overall chi-square total for the K-Stage model. The large values early in the interval result from division by the very small expected number of incidents for these days. Using the chi-square values to evaluate model fits is useful, but should be interpreted with caution for this data. Unlike typical software reliability data, the incident data includes some uncertainty regarding the start of the measurement interval and contains days where very large increases in the number of incidents are observed. The chi-square results for the incident data need to be reviewed in this context.

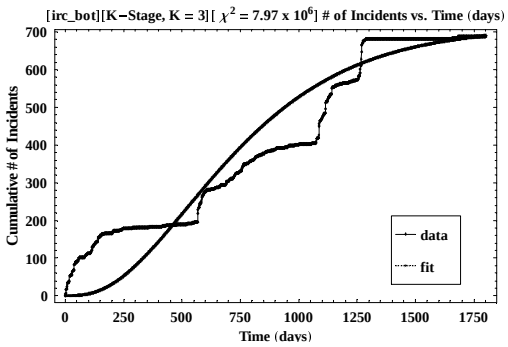

Figure 6.19: K-Stage model fit and data for "irc_bot".

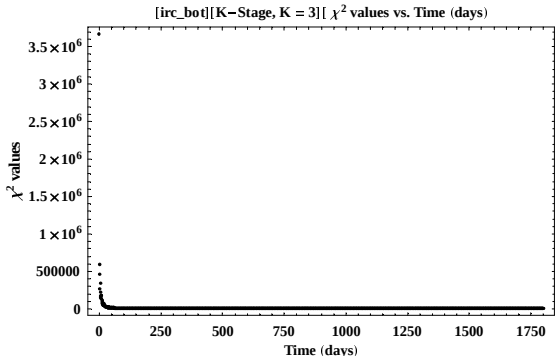

Figure 6.20: Chi-square residuals for K-Stage model fit of "irc_bot" data.

Since the interest is in developing models to be used for forecasting, model selection can be based on the accuracy of model predictions for unseen data. This is evaluated by splitting the data and using the hold-out cross-validation technique [87] for comparing models. Using this method, some of the data is withheld and not used for estimating parameters and then models are used to forecast values which are compared to the withheld data. An advantage of using this method for model selection is that it reduces the risk of selecting models which may "overfit" the data (i.e. a model that only provides a good fit to the data in the range used for estimating parameters and provides a poor fit for data outside this range). This is an important selection consideration for models that will be used to forecast beyond the range of the currently available data.

Table 6.3 shows how the data was split for each incident type. Table 6.4 shows the chi-square fit values for both the data used for model parameter estimations and for the hold-out data. Figures 6.21-6.24 show some examples of model fits to parameter estimation data and hold-out data. Table 6.4 shows the G-O model created using the training data for parameter estimation best fits the held-out "worm_msblast"

Incidents	[Full Set] Days included	[Full Set] # of incidents	[Training] Days included	[Testing] Days included	[Training] # of incidents	[Testing] # of incidents	[Training] # days w/ incidents	[Testing] # days w/ incidents
All data	2100	11966	1400	700	9692	2274	901	412
worm_msblast	32	2133	21	11	2033	100	15	8
virus_generic_bot	1008	1940	672	336	1550	390	225	106
virus_klez	659	1118	439	220	1083	35	168	19
bagle_worm	679	849	452	227	828	21	69	10
irc_bot	1802	690	1201	601	565	125	220	30
virus_agobot	702	589	468	234	588	1	66	1
rogue_ftp	1756	500	1170	586	467	33	204	25
nethicsreq	1949	509	1299	650	374	135	157	56
spamrelay	1733	429	1155	578	191	238	110	128
worm_nachi	199	317	132	67	308	9	24	6

Table 6.3: Splitting incident data for hold-out validation.

incident data when compared to other models. The K-Stage model created using the training data provides the best fit to the held-out "virus_agobot" incident data. This is more consistent with the Laplace trend analysis for this data than the chi-square fit values shown in Table 6.2 for models with parameters estimated using the full data interval. Table 6.4 also shows that although the K-Stage model does not provide the best predictions for the "irc_bot" incident data, it is not the worst predictor as one might conclude from the chi-square fit values in Table 6.2. The Duane model predicts the "spamrelay" incident values better than the other models. Trend analysis showed a decreasing to stable reliability growth trend which typically means the G-O model would perform poorly (as indicated in Tables 6.2 and 6.4). The Duane model is unique from the other reliability growth models examined in this paper as it was derived from the hardware reliability area where the other models were developed from software reliability observations.

Laplace trend information and plots of data and model fits are not shown for the "worm_nachi" incident data, but is similar to what is shown for "worm_msblast."

Incidents	[G-O] [Training] X^2 values	[S-shaped] [Training] X^2 values	[K-stage] [Training] X^2 values	[Duane] [Training] X^2 values	[G-O] [Testing] X^2 values	[S-shaped] [Testing] X^2 values	[K-stage] [Testing] X^2 values	[Duane] [Testing] X^2 values
All data	820,773	151,563	187,151	162,992	90,866	234,888	49,126	590,994
worm_msblast	466	1,876	5,875	2,146	8	15	16	377
virus_generic_bot	21,841	23,123	83,239	33,908	145	4,406	8,332	5,475
virus_klez	17,487	114,529	542,605	4,084	45	84	148	2,215
bagle_worm	9,783	10,122	14,689	32,504	20	26	26	6,573
irc_bot	30,761	289,602	7,329,742	16,483	4,376	1,778	4,323	1,963
virus_agobot	27,456	23,812	116,668	25,047	1,206	159	29	34,363
rogue_ftp	27,834	44,220	1,705,429	17,823	11,280	11,925	4,200	21,468
nethicsreq	19,370	10,305	366,672	12,807	4,283	730	806	12,691
spamrelay	19,092	13,487	2,022,188	11,514	17,225	1,987	8,441	915
worm_nachi	547	5,634	37,744	903	1.4	1.6	1.6	1,094

Table 6.4: Chi-square fit values for split incident data.

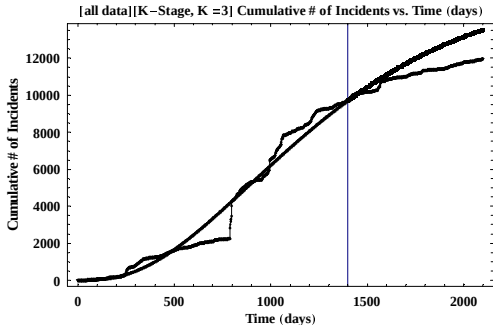

Figure 6.21: K-Shaped model fit and data with hold-out data included for "All data".

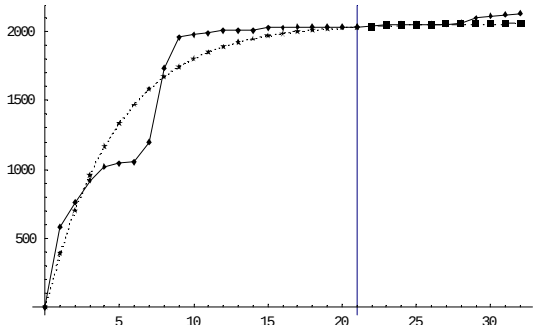

Figure 6.22: G-O model fit and data with hold-out data included for "worm_msblast".

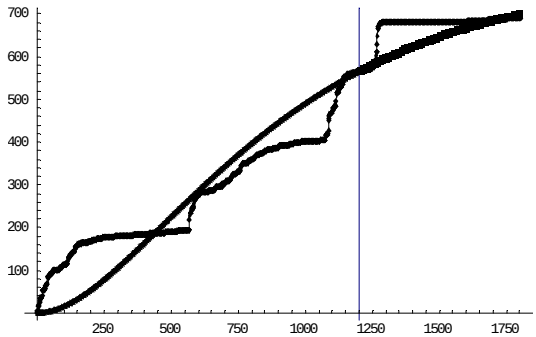

Figure 6.23: S-Shaped model fit and data with hold-out data included "irc_bot".

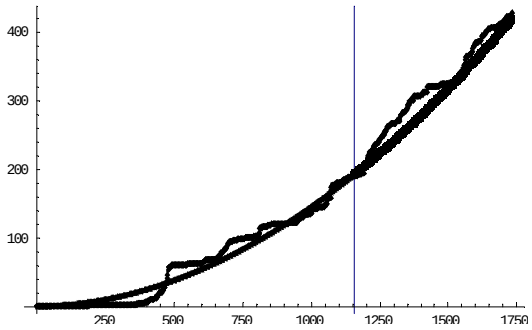

Figure 6.24: Duane model fit and data with hold-out data included for "spamrelay".

This is expected as the Blaster and Nachi worms exploited similar vulnerabilities and occurred during a similar time frame.

6.5.2 Time series models

The use of time series modeling is explored for a range of computer security incident data. As previously discussed, several incident types may spread due to factors directly analogous to those affecting the spread of contagious diseases in animals. The occurrences of other incident types are potentially more influenced by factors not included in the infectious disease based models. Since time series models do not attempt to directly account for or use information about causal factors, they can be applied to a broader range of incident types.

Figure 6.25 shows the cumulative values for the "spamrelay" data and associated sample ACF values for $d = 0$ and $d = 2$. Also shown is a once differenced series of the log transformed weekly values and corresponding ACF plot. We see that there is a high amount of autocorrelation present in the cumulative values indicating that this data is nonstationary. Differencing the cumulative data ($d = 2$) appears to stabilize the mean. Applying the log transformation appears to reduce some of the variance. The actual transform was $log[x + 1]$ because there are weeks in which no incidents occurred. We explored the use of this transformation for all of the top incident types.

Models were calculated for combinations of p and q values each ranging from $0 - 5$ and the models with the lowest AICc and BIC values were identified. (There

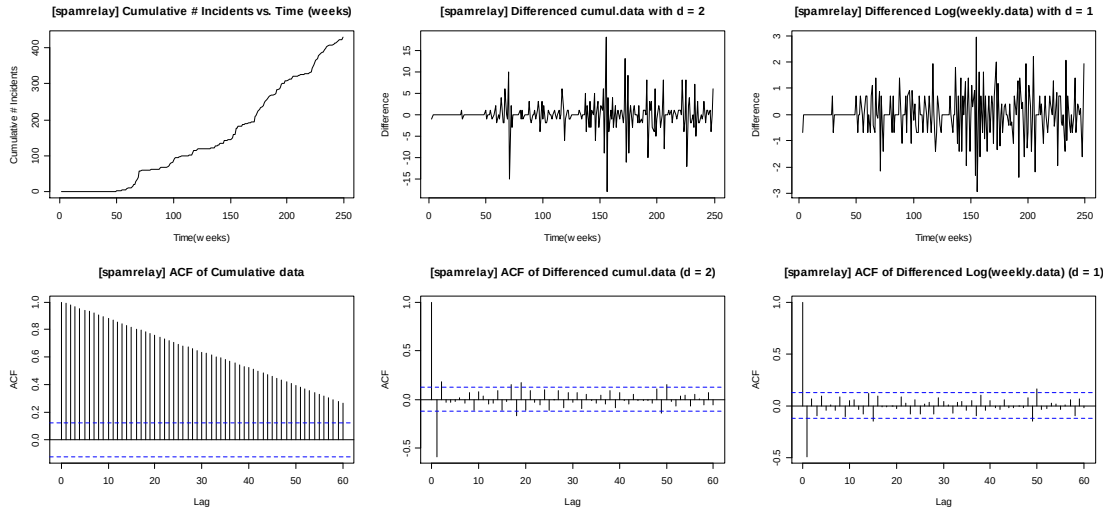

Figure 6.25: Difference transformations and ACF values for "spamrelay".

are methods for evaluating models with AICc or BIC values within a certain range of closeness to the lowest values, however, these methods can involve some subjective judgments and were not explored.)

6.5.2.1 Fitting and forecasting observed data with time series models

A basic modeling process is followed. First, transformations (differencing, log) are applied to the data to stabilize the mean and variance. Next, models are generated for a range of parameters (p, q). From the generated models, models are selected based on lowest AICc and BIC values.

The interest is in developing models for forecasting the number of computer security incidents in a future time interval, so the accuracy of model predictions are compared to unseen incident data. This is done by splitting the data and using the hold-out cross-validation technique [87] for comparing model predictions. Using this method, some of the data is withheld and not used for parameter estimations or

model selection. Then model forecast values are compared to the data that have been withheld. Model parameters are estimated using maximum likelihood method with the R statistical software package [68]. An example of the R code used is provided in Appendix D.

A decision was made to split the data by time (instead of by number of incidents). It is of more interest to examine the expected trends over time instead of forecasting when a specific number of incidents will have occurred. Data from the first two-thirds of the time interval was used for model order selection and model parameter estimation. The remaining data was used for forecast validation. This is the same method used when estimating software reliability growth model parameters and selecting models.

In addition, some multiplicative seasonal ARIMA (SARIMA) models are examined. A one year period was used when possible to explore possible influences of the academic calendar schedule on occurrences of incidents. Since the length of the data interval being modeled must be at least twice the length of the seasonal period to allow for differencing, not all of the incident types could be modeled using a 52 week period after being split into modeling and validation sets. A period of 26 weeks was used for incidents with shorter intervals of observation because it is a factor of 52 and provides some overlapping of spring and fall semester activity. While it seems reasonable to expect some incident types might be affected by the academic calendar (observed occurrences increase during spring and fall semesters), it is difficult to capture using a single lag offset value due to variations in the lengths of semester breaks. In the case of the "nachi" incident type, a period of 5 weeks was

Incident Type	Selection Criterion	Not Seasonal		Seasonal		Seasonal Period Weeks
		Cumulative	Log weekly	Cumulative	Log weekly	
		Forecast Data RMS	Forecast Data RMS	Forecast Data RMS	Forecast Data RMS	
all data	AICc	1418.04	**224.59**	2104.02	869.32	52
msblast	AICc	1333.48	53.31	n/a	n/a	n/a
genbot	AICc	182.01	117.50	220.93	**36.46**	26
klez	AICc	66.93	86.61	**43.54**	53.43	26
bagle	AICc	72.62	**6.35**	220.03	17.86	26
ircbot	AICc	**80.58**	**58.20**	153.43	93.26	52
agobot	AICc	21.59	1.96	1.58	5.46	26
rogueftp	AICc	124.22	52.67	767.64	34.60	52
nethicsreq	AICc	27.96	**41.09**	137.11	**63.47**	52
spamrelay	AICc	72.35	**111.08**	**35.25**	55.66	52
nachi	AICc	**4.14**	4.03	27.43	7.61	5

***Shaded** cells indicate lower RMS values compared to BIC selected models*

Table 6.5: ARIMA models and forecasts (AICc based selection).

chosen due to the shorter time interval of the data. A period of 5 weeks may capture some monthly influences if present. Closer examination of the ACF plots for each incident type may also indicate other seasonal periods to try.

Tables 6.5 and 6.6 show the models selected for each incident type based on lowest AICc and BIC values. AICc and BIC are widely used information criteria for model selection, but alternate selection criteria have been proposed such as the Hannan-Quinn information criterion (HQC) [88]. Model selection was independent of the data withheld for validation. The root-mean-squared (RMS) errors were calculated to compare forecasts from the selected models to data withheld for validation. Table 6.5 shows results of the AICc selected models using the cumulative values and log transformed weekly values, while Table 6.6 shows the results for the BIC selected models. The forecasts for the log transformed weekly value models were converted to cumulative values for comparison purposes. The "msblast" incident type did not have enough weeks of data to calculate seasonal models.

We can compare how the choice of model selection criteria affects forecasting.

Incident Type	Selection Criterion	Not Seasonal		Seasonal		Seasonal Period Weeks
		Cumulative	Log weekly	Cumulative	Log weekly	
		Forecast Data RMS	Forecast Data RMS	Forecast Data RMS	Forecast Data RMS	
all data	BIC	**	273.14	**	**	52
msblast	BIC	**1222.72**	**	n/a	n/a	n/a
genbot	BIC	**	**108.99**	**	65.37	26
klez	BIC	**	**	430.62	**	26
bagle	BIC	**	8.90	**	**	26
ircbot	BIC	84.76	93.79	**72.54**	**88.14**	52
agobot	BIC	**0.17**	**0.17**	**	**	26
rogueftp	BIC	*	**48.48**	**	**	52
nethicsreq	BIC	**26.15**	41.23	**	65.24	52
spamrelay	BIC	**	120.74	35.44	**54.27**	52
nachi	BIC	20.03	**	**	**6.86**	5

*** indicates same model selected using AICc, see Table 6 for RMS value*
***Shaded** cells indicate lower RMS values compared to AICc selected models*

Table 6.6: ARIMA models and forecasts (BIC based selection).

In half of the cases (21 out of 42), AICc and BIC select the same model. Of the remaining 21 cases where AICc and BIC select different models, using AICc resulted in picking a better predictive model 11 times, while using BIC resulted in better predictive models in 10 instances. Simply using AICc to select models did slightly better than using BIC. If overfitting is a concern, another strategy would be to examine the models selected using AICc and BIC and then choose the one of lower order. Although the model orders are not shown in Table 6.5 or Table 6.6, this strategy would not have improved upon using only AICc for model selection.

Comparing the non-seasonal models constructed using the cumulative values to the ones using the log transformed values shows improved forecasting performance using the log values in eight out of eleven incident categories. Overall, it seems using the log transform improves forecasts in more cases than it degrades forecasts. The magnitude of forecasting improvements also appears to be larger than most forecasting degradations that occur.

Including a seasonal element into the models improved forecasts using the cumulative data for three incident types: "klez", "spamrelay", and "ircbot", while for the log transformed values it improved forecasts for four incident types: "genbot", "klez", "rogueftp", and "spamrelay". Other than "klez", the incident types benefiting from including a seasonal factor are named after an observed function of a compromised machine and these types are not easily associated with a particular piece of malware or exploited vulnerability. Incidents in these categories could potentially result from several different underlying causes yet exhibit the same outward behavior. The frequency of these incident types are more likely to be influenced by the number of active machines on the campus network than the exploitation of a particular vulnerability.

Some forms of malware are modular in nature and have multiple variants. These variants can incorporate exploits for new vulnerabilities as they are discovered. Malware of this type often provides some desired functionality to an attacker (such as relaying spam or becoming part of botnet to be rented out for distributed denial-of-service attacks or phishing schemes). These incident types may persist even as specific vulnerabilities are patched, as long as an economic incentive exists for the desired functionality provided.

Besides academic calendar, another potential source of periodic behavior could result from monthly patch cycles used by some vendors. To examine this, the data could be grouped by months and scaled to account for differences in the number of days per month. This could be applied to the complete set of incidents as well as some individual incident types.

6.5.3 Comparing software reliability growth and time series models

Software reliability growth (SRG) models are often used to describe and predict failures and faults in software systems. In many systems, reliability is assumed to increase over time as faults are found and fixed and as users become more familiar with the features of a system and can avoid or adjust to different failure states. This often leads to an observed decreasing rate of failures.

While security incident data is not the same as software failure data in systems, there are some similarities. A previous study explored the concept of viewing computer security incidents on a network as analogous to software errors in a system and tested the prediction ability of some common SRG models [89]. time series models have also been applied to modeling SARS epidemic data [61]. Another study (involving several small sets of data from different types of repairable systems) compared forecasts from ARIMA time series models to forecasts from an SRG model and found the time series models forecasts to be similar in most cases and slightly better in a few instances [90].

The comparisons provided in this section are based on the four different SRG models previously discussed: the Goel-Okumoto (G-O) model, the S-Shaped model, the K-Stage Curve model, and the Duane model. These models were selected because they represent some of the common NHPP models in use today.

Table 6.7 shows a comparison of the best time series model forecasts to the best SRG model forecasts for each incident type. The best SRG model yields better forecasts than the best selected time series model in seven of eleven cases.

Incident Type	TS	SRG
all	**224.59**	937.44
msblast	53.31	**6.14**
genbot	36.46	**26.89**
klez	43.54	**15.36**
bagle	**6.35**	8.44
ircbot	58.20	**43.00**
agobot	**0.17**	8.90
rogueftp	**34.60**	64.67
nethicsreq	26.15	**22.05**
spamrelay	35.25	**19.51**
nachi	4.03	**3.88**

***Shaded** cells indicate lower RMS values*

Table 6.7: Comparison of best model forecast RMS values for time series and SRG models.

This result suggests viewing computer security incident data (when separated into different types) as similar to a class of detected errors in a software system can be a useful approach for monitoring and forecasting. However, when modeling the total aggregate number of incidents over time (resulting from the combination of all incident types) a time series model may be more appropriate. This is not surprising as even if factors affecting the propagation and occurrence for each different incident type were well understood, the intervals at which new incident types are found is likely a less understood process and better captured by time series models.

Of the individual incident types modeled, the "spamrelay" incident type is unique in that it displays a slight increasing trend over time and the best fit was the Duane model. This continued increasing trend probably results from economic incentives behind attacks to compromise machines to use for sending and relaying spam. These incentives drive innovation to develop new techniques for compromising

machines for this purpose.

In some cases, applying a single SRG model across a whole interval may not be appropriate. The observed "agobot" incidents mostly fall into two distinct groupings– a small group at the beginning and much larger group about a third of the way into the interval. Only a few occurrences are observed after week 40. Agobot is an example of modular malware that can easily be modified to include new exploits. The second grouping of incidents likely results from a different exploited vulnerability than the first group. If so, it would be more appropriate to apply SRG models to each group. The data does not identify different variants of the agobot malware.

The most frequently observed incident types were examined. These incident types likely attracted enough attention for some form of intervention and control measures to be taken to reduce the number of occurrences. As each measure has some cost involved (labor, equipment, inconvenience to users, etc.), being able to gauge the effectiveness of the measures is important. Comparing model forecasts to actual observations before and after implementing a control measure can provide valuable feedback regarding a control measure.

6.5.4 Update regarding "spamrelay" incident type

While many of the incident types examined have a "closed" time interval of occurrence (a date after which no more incidents of the type are observed), the "spamrelay" incident type continued to be observed after the original SRG and TS model analysis was performed. This provides an opportunity to follow-up with

additional observations and provide additional insight regarding this incident type relative to the campus environment.

The "spamrelay" incident type classification reflects only the observed behavior of a compromised computer and is not related to a particular type of vulnerability or exploit. This means there are many possible ways an affected computer could have been compromised and there is unlikely to be a single patch or update that can be applied that would effectively address this incident type. The growth of this incident type displayed during the original time interval analyzed (June 2002 to March 2007) is likely due to the increase of economic incentives as a driving motivation behind more types of computer security incidents [91]. In particular, resources (such as verified email address lists and open mail relays) related to the practice of spamming have been observed on underground markets dealing in "illicit digital goods and services in the support of criminal activities" [92].

One of the purposes of control type models (such as SRG models) is to provide feedback regarding a process and whether or not a particular action or intervention has a measurable impact. This can be useful for examining if a particular policy change or intervention has an effect on an incident type which may not have easily identifiable vulnerabilities or exploits to address (such as "spamrelay").

Figure 6.26 shows the Duane model projection and actual incident data for an additional time interval (April 2007 to April 2010). The Duane model projection is still based on the same original training data time interval. As can be seen in Figure 6.26, the numbers of observed incidents are noticeably fewer than the model's projected values in the additional time interval. A possible explanation for this

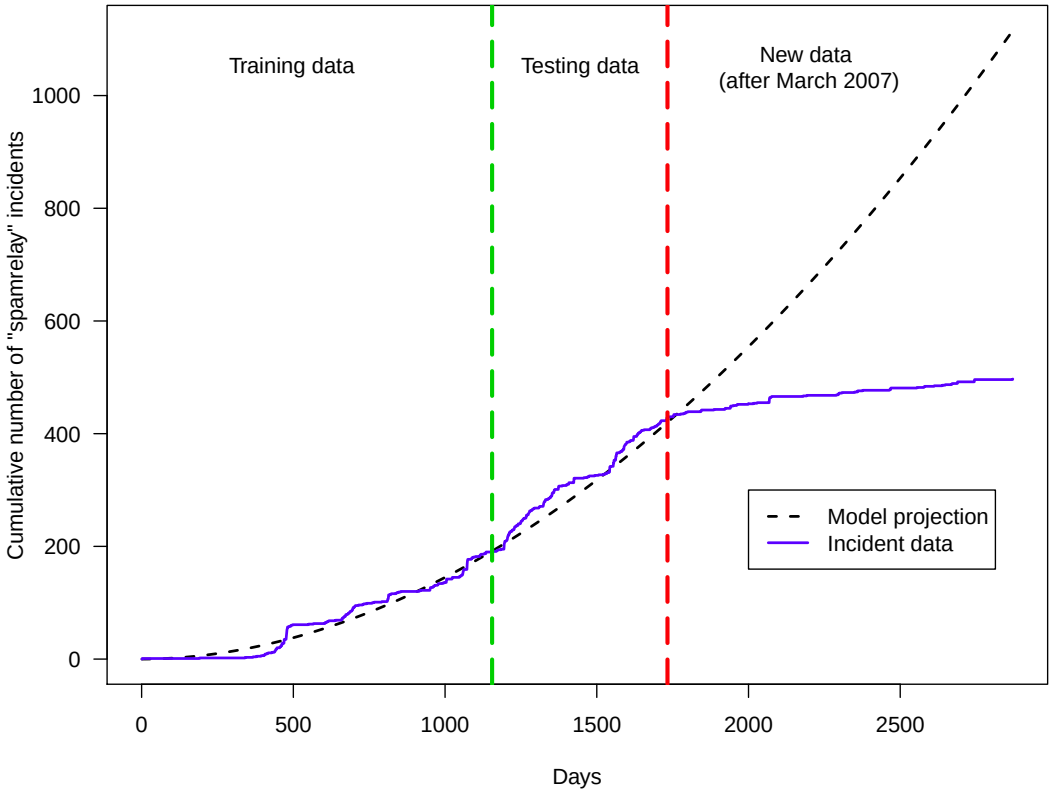

Figure 6.26: Duane model fit and incident data for "spamrelay" with additional time interval.

difference is that during the spring and summer of 2007, the campus consolidated many disparate email systems and accounts to a centrally managed system. Email servers on the campus borders could then be configured to be more selective regarding which internal campus IP addresses outgoing mail would be accepted from rather than being configured to accept outgoing mail from any internal campus IP address. This change reduced the value of a compromised internal campus computer for the purposes of relaying spam through the campus mail servers to external destinations.

However, while these changes in the campus email environment may have reduced the value and therefore the incentive to compromise computers for the purposes of relaying spam, the changes also likely increased the value of compromised campus email accounts for purposes of sending spam through the campus mail servers (since the validation of outgoing mail shifted more to a valid campus email account away from a valid campus IP address). An analysis of successful campus email account phishing attacks in 2008 is presented in [93] and an increase in these types of attacks could partly be in response to changes to the campus mail environment.

This section illustrates the use of an SRG model to potentially identify the impact of a change in an email environment on the occurrence of the "spamrelay" incident type. Since the occurrence to this particular incident type is most likely a reflection of the economic incentives attackers have for this resource, an effective intervention is likely to be one that either reduces the value of this resource and/or increases the cost to attackers for obtaining this resource.

This chapter illustrates the application of several different types of models to a set of computer security incident data. Some types of models are used for gaining a

better understanding of the factors and dynamics relevant to incident occurrence. A better understanding of these factors and dynamics can lead to better interventions and focusing of intervention efforts. Other types of models are used for developing forecasts of future levels of incident occurrence. These models can be used to evaluate the impact of interventions or policies on incident occurrence, even when the relevant underlying factors or dynamics are unknown or not easily modeled.

Chapter 7: Exploring Population Risk Factors

Previous chapters cover some models which incorporate factors that could potentially influence the spread of worms or viruses. These factors include the distribution of individual likelihoods to open infected email messages, the number and structure of social contacts that connect the nodes in a network, and assumptions regarding the prevalence of software vulnerabilities and the rate at which vulnerabilities are patched. Although these may all be important and influential factors, they can also be difficult to directly measure or quantify. However, other features may be more easily measured or estimated and some of these features may either correlate or in some way function as a proxy measure (or indicator) for factors included in previously covered models. If some of the observable factors are associated with an increased risk of incidents, then they may provide an opportunity for targeting or focusing intervention efforts (such as patching, blocking, or user-awareness training).

7.1 Description of data

In this chapter, we use the full range of incident data as we are not using it to model individual incident types. Instead, we are exploring relationships between population characteristics and computer security incidents of all types. In addition to

the incident data described in the previous chapter, the following data was also made available. Log files from some of the network routing equipment on campus provided monthly address resolution protocol (ARP) cache information. As collected, the ARP cache information indicates when a particular IP address and MAC address pairing was first and last seen in use during the month (the amount of actual time connected between the first and last seen timestamps cannot be determined conclusively from this information). A recorded IP address and MAC address pairing may have been in use continuously or intermittently between the first and last seen timestamps.

An initial intention was to examine differences between several campus subpopulations and grouped the campus subnets by colleges and administrative units. However, because the dominant observable difference in incidence rates was between hosts on the housing networks (dorms, on-campus apartments, residence halls) and hosts on any other part of the campus network (academic departments, research labs, and administrative units), the categorization of subpopulations was simplified to "Housing" and "Other." Dial-up, wireless, VPN and other dynamically addressed networks are excluded from either category.

To calculate incidence rates that could be compared across groups of different size, the number of incidents was divided by the number of "host-months" where a "host-month" is defined as the presence of one unique IP/MAC address pair on the campus network at any time during a particular month. The number of "host-months" for any given month is used as the population count of hosts on the campus network for that month. A unit of "host-month" does not differentiate between a computer connected to the campus network for the entire month vs. a computer connected to

the campus network for only a few days that month.

The time interval between the first and last seen timestamps for each IP/MAC address pairing was calculated. Only pairings with time intervals of 24 hours or greater were retained. This was done because in some cases the router caches retained IP/MAC address pairings for a few hours even if the actual usage was only for a few minutes. Pairings with intervals of less than 24 hours were discarded to increase the likelihood that an IP/MAC address pairing was actively being used on the network instead of appearing in the ARP cache data as the result of configuration errors.

It is possible for more than one MAC address to be associated with the same IP address within a month. In some cases, this may be caused by misconfigurations where two (or more) machines attempt to use the same IP address at the same time. In other cases, the same IP address may be used by several machines but only at different times such that no more than one device with that IP address is actually connected to the network at any time. From the recorded timestamp information alone, it is not possible to determine when each different device was used on the network in a given month if there are overlapping timestamps. In these cases, the MAC address with the longest time interval between the first and last seen timestamp was retained as being associated with the IP address while other MAC addresses associated with the same IP address were discarded for that month. MAC addresses were also checked and limited to being associated with a single IP address per month. There is a one-to-one correspondence between the MAC addresses and IP addresses retained for each month.

The monthly data are grouped by academic calendar year. The starting month

is defined as August and the ending month as the following July. There were 11,979 incidents in the original data. After removing IP/MAC address pairs from the ARP cache data, there were 9,334 incidents with IP/MAC address pairs remaining in the data for the month the incident was recorded. The percentage of original incidents retained per academic year ranges from 67% to 87% with an overall average of 78%.

7.2 Population Characteristics

The data includes information about the following population characteristics. Each characteristic is examined to look for associations with the observed computer security incidents. These associations may not be causal in nature, yet may be useful for identifying machines more at risk of being involved with an incident.

7.2.1 Network affiliation of host (Housing/Other)

This population characteristic identifies if a particular IP/MAC address pair is part of the housing subnets on-campus ("Housing") or part of other subnets ("Other"). Table 7.1 shows the total number of host-months per academic year for each group as well as the number of incidents and the incidence rate (IR) per 1,000 host-months. The last column in Table 7.1 shows the ratio of [Housing IR]/[Other IR]. Except for 2001/02, the incidence rate on the "Housing" networks has been from 2 to almost 10 times higher than the incidence rate on the "Other" networks for the same academic year.

Potential reasons for the difference in incidence rates between the two groups

Table 7.1: Incidence rates for Housing and Other networks.

Academic Year	Housing Host-months	Housing Incidents	Housing IR per 1,000 host-months	Other Host-months	Other Incidents	Other IR per 1,000 host-months	Ratio of [Housing IR]/ [Other IR]
2001/02	110,123	133	1.21	190,910	242	1.27	0.95
2002/03	113,013	569	5.03	202,291	162	0.80	6.29
2003/04	117,214	2,345	20.01	201,598	2,003	9.94	2.01
2004/05	124,720	1,337	10.72	204,491	357	1.75	6.14
2005/06	124,631	799	6.41	212,456	212	1.00	6.42
2006/07	133,110	481	3.61	209,851	188	0.90	4.03
2007/08	96,112	230	2.39	213,373	52	0.24	9.82
2008/09	53,638	80	1.49	215,304	42	0.20	7.65
2009/10	46,973	53	1.13	199,169	49	0.25	4.59

could be due to differences related to user behavior or how the computers are used. Or, there could be differences related to characteristics of the computers, such as how current the operating system is or whether the machine is always on and connected to the network. Other available information is used to examine if there are identifiable and measurable factors that may account for some of the differences in the incidence rates of these two groups.

7.2.2 Age

For our analysis, we define the "age" of a host as the time in months (plus one) between when a particular MAC address was first seen on the campus network and any particular month in the data that the same MAC address was present (independent of its IP address). Since identification of individual machines was based on MAC addresses and not IP addresses, the "age" of a host was always calculated based on the month its MAC address was first seen on the campus network, regardless if its IP address has changed.

This variable was included as it may be associated with the age and/or patch

Table 7.2: Incidence rates for Low and High aged machines.

Academic Year	Low-aged Host-months	Low-aged Incidents	Low-aged IR per 1,000 host-months	High-aged Host-months	High-aged Incidents	High-aged IR per 1,000 host-months	Ratio of [Low-aged IR]/ [High-aged IR]
2001/02	158,981	233	1.47	142,052	142	1.00	1.47
2002/03	165,755	507	3.06	149,549	224	1.50	2.04
2003/04	162,173	2,720	16.77	156,639	1,628	10.39	1.61
2004/05	167,980	1,261	7.51	161,231	433	2.69	2.80
2005/06	177,374	698	3.94	159,713	313	1.96	2.01
2006/07	179,398	370	2.06	163,563	299	1.83	1.13
2007/08	157,269	206	1.31	152,216	76	0.50	2.62
2008/09	137,280	92	0.67	131,662	30	0.23	2.94
2009/10	124,547	56	0.45	121,595	46	0.38	1.19

level of a computer's operating system (and/or other installed software). Older machines may be running older operating systems (or software) which contain more identified vulnerabilities and increase their risk of being involved in an incident (for incident types that target vulnerable software). It is also possible that older machines still connected to the network have been hardened in some way or run less vulnerable operating systems so they may have a decreased risk of being involved in an incident. On the "Housing" networks, older machines may also be associated with older students who may have different usage behaviors.

Table 7.2 shows the incidence rates per academic year for machines grouped into two age categories–"Low" and "High." For each month within an academic year, the median age of the machines was identified and machines with ages less than or equal to the median age were categorized as "Low"; machines with ages greater than the median age were categorized as "High." The number of incidents for machines in each category was also counted for each month. The counts were done on a monthly basis to minimize the effect of threats and risks that may change over the time period of the academic calendar year. For example, if a particular incident type exploits several machines in October of a particular academic year, it may appear

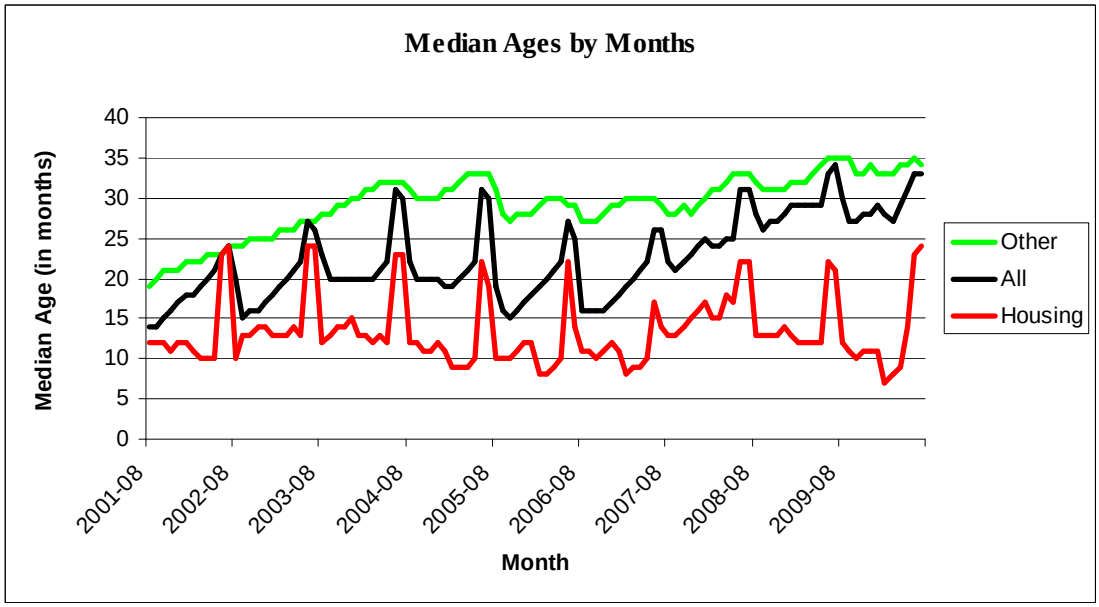

Figure 7.1: Median ages by months.

that machines that are three months old are more at risk of incidents. However, there may just be more three-month-old machines in October because many new machines are connected to the campus network sometime in late August (or the typical start of the academic year). Of more interest is examining the following: of the machines on the campus network in October, is there a difference in risk of incidents related to the ages of the machines present? Table 7.2 indicates that for each academic year, the incidence rate of "Low" machines ranged between 1.2–2.8 times higher compared to the incidence rate for "High" machines in the same academic year.

One possible reason for difference in incidence rates between "Low" and "High" machines is that there may simply be different distributions of "Low" and "High" machines on the "Housing" and "Other" networks. Figure 7.1 shows the monthly median ages for the entire population as well as for the housing and other groups. It can be seen there is often a 15-20 month "age" difference between the median ages

of the two populations. It should also be noted that since the ARP cache data used to calculate ages only goes back to January 2000, the ages are underestimated for some of the population. The effect of this underestimation is mostly seen in the first few years and is visible as the steady increase in median age for "Other" between 2001 and 2004.

7.2.3 Calendar month

Next the data are grouped by calendar month to see if there is any seasonal variation present. For this analysis, a scaled or proportional incidence rate is used that is calculated by dividing the proportion of incidents per calendar month per academic year by the proportion of hosts per calendar month per academic calendar year. This was done to allow each year to have equal weight and avoid cases where calendar months in years with more incidents are weighted more that those in years with fewer incidents. For example, if for a particular academic year, the month of January had 10% of all the incidents for that academic year and January also had 10% of all the host-months for that year, then it would have scaled IR value of 1.0. A scaled incidence rate value was calculated for each month of each academic year and Figure 7.2 shows boxplots of the scaled values for each calendar month. The figure on the left shows the boxplots for values calculated using all of the data, while the other two boxplots are for "Housing" network hosts only (the middle plot) and "Other" network hosts only (the plot on the right).

The pattern of the boxplots is similar for the different populations (all, housing,

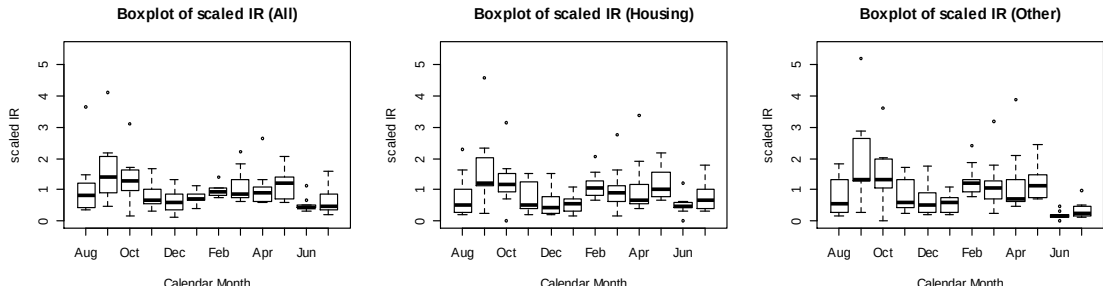

Figure 7.2: Boxplots of yearly scaled incidence rates by Calendar Month.

other) where proportionally there are more incidents that occur in September and October and the fewest in June.

7.2.4 Academic semester

In addition to looking at the separate calendar months, the data are also grouped into designations that align with parts of the academic calendar. Three designations are used: Fall (August-December), Spring (January-May), and Summer (June-July). We use the same approach for calculating a scaled incidence rate for each semester designation of each academic year as described in the previous section for Calendar Month.

Figure 7.3 shows the scaled incidence rates of each designation for each academic year of data. While for many years in the middle, the Fall semester had proportionally more incidents and the Summer season had proportionally less, in more recent years the proportionally scaled incident rates have become more similar for the different semester categories. Figures 7.4 and 7.5 show similar patterns for the "Housing" and "Other" networks, but there is a little more separation between the Fall and Summer rates for "Other" compared to "Housing."

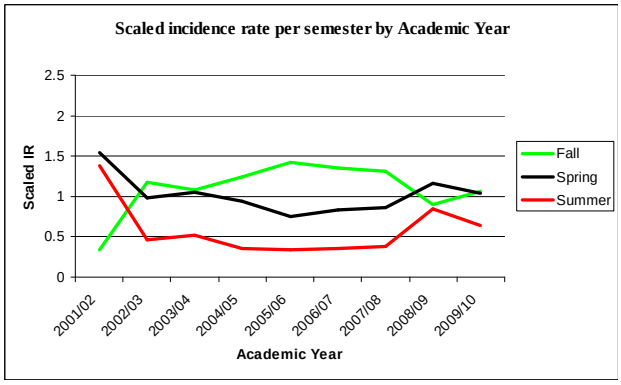

Figure 7.3: Scaled incidence rates per semester by Academic Year.

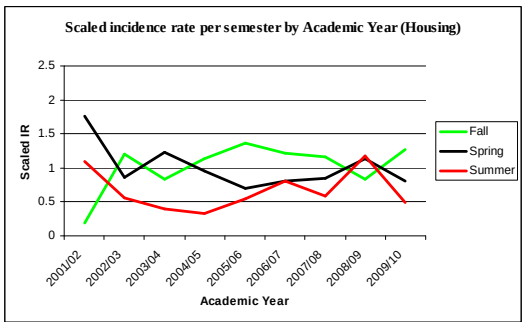

Figure 7.4: Scaled incidence rate per semester by Academic Year (Housing).

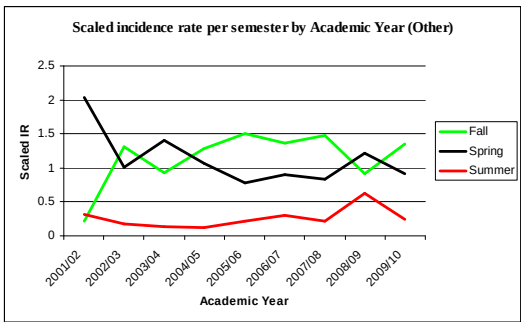

Figure 7.5: Scaled incidence rate per semester by Academic Year (Other).

7.2.5 Vendor ID from MAC address

From the ARP cache data, the MAC addresses of machines on the network can be obtained. The first part of the MAC address identifies the vendor or manufacturer of the network interface. Three groups (Apple, Dell, Other) are designated based on the available vendor ID part of the MAC (or Ethernet) addresses of hosts present on the campus network. Although the relative number of hosts with Apple Ethernet addresses was comparatively low (from 5.5% in 2003/04 to 10% in 2009/10) there is likely a strong correlation between hosts with Apple Ethernet addresses and hosts running an Apple operating system (such as OS X). If a host's operating system affects its risk of being involved with a security incident, this designation may capture some of this association. The most prevalent MAC address vendor ID was Dell which is why it was used as a separate category. Although only 15% of hosts in 2001/02 had Dell Ethernet addresses, in 2009/10 this number was over 52%. Remaining Ethernet address vendor IDs were grouped into Other (as opposed to "Other" which is used for the Housing/Other subnet groupings).

Figure 7.6 shows the scaled incidence rate (calculated similarly as described for Calendar Month) for the three different MAC address vendor ID categories. The most notable feature in Figure 7.6 is the increasing trend seen in the Apple category. This increase means the proportion of incidents involving hosts with Apple Ethernet addresses has been increasing faster than the proportion of hosts with Apple Ethernet addresses connected to the campus network in general. This same trend is seen in Figure 7.7 and Figure 7.8 which show the scaled incidence rates for the "Housing"

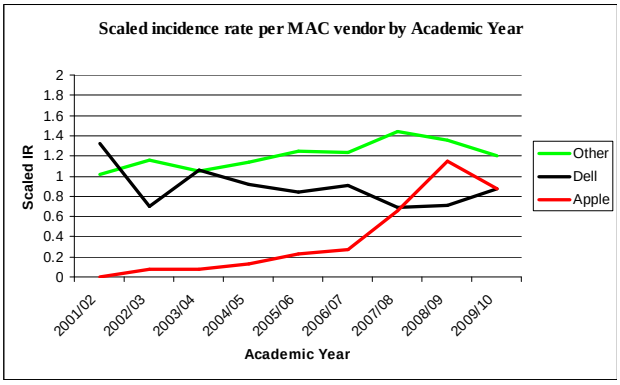

Figure 7.6: Scaled incidence rate per MAC vendor by Academic Year.

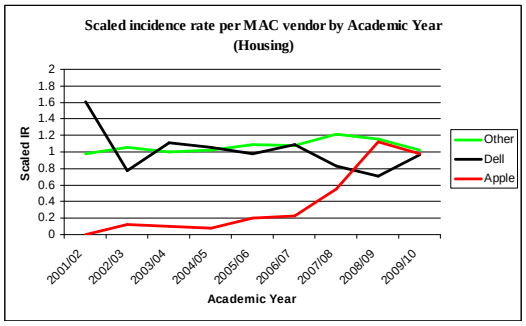

Figure 7.7: Scaled incidence rate per MAC vendor by Academic Year (Housing).

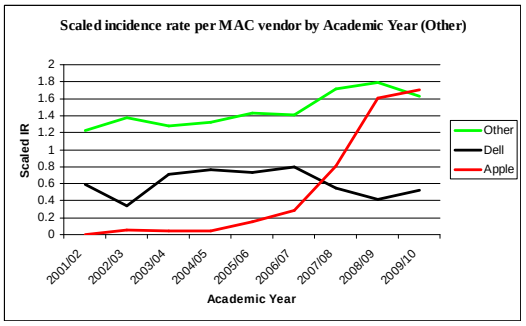

Figure 7.8: Scaled incidence rate per MAC vendor by Academic Year (Other).

and "Other" network populations. One difference between Figures 7.7 and 7.8 is there is a larger separation between the Dell and Other categories in the "Other" network population than in the "Housing" network population. One potential reason for this could be that many of the Dell hosts in the "Other" network population group may be part of large institutional purchases and therefore more likely to be centrally managed and maintained (and therefore having a lower risk of being involved in an incident), while most Dell hosts present in the "Housing" network population group may be individually owned and managed and have more similar risk as "Other" machines.

7.2.6 Network time

When calculating incidence rates, the number of "host-months" has been used as the denominator which gives equal weight to each host seen on the campus network each month. However, other denominators could be used. One possible denominator could be expressed as host-network time, where network time would reflect the amount of time a host was connected to the campus network. This would reflect a difference between a host that was present and connected for a few days compared to a host that was present and connected everyday. Using the ARP cache data, the difference in time is calculated between when an IP/MAC address pair was first and last seen for a given month. However, there is some uncertainty involved with this measure as the status of the machine in between these two timestamps is unknown–two hosts that were connected for the same number of days would have the same value of network time even if one was left on all the time and one was turned off each night. The reason for including network time is that some incident types (such as worms that spread directly from machine to machine without user action) may affect hosts that are connected for longer periods of time than hosts that are connected for shorter periods of time. The counter-argument to this is that some incident types require a user action to occur and what really matters may be the amount of time a user uses a particular host instead of just how long a particular host is on and connected to the network.

To examine the effect of network time, four equal-sized time categories (Q1, Q2, Q3, Q4) are defined with boundaries determined by splitting a 30-day month

Table 7.3: Incomplete timestamp data.

Academic Year	Months with incomplete data
2001/02	2001-Sep
2002/03	2002-Sep, 2002-Nov, 2003-Mar, 2003-Apr
2003/04	2004-Mar
2004/05	-
2005/06	-
2006/07	2006-Aug, 2006-Sep
2007/08	-
2008/09	-
2009/10	2009-Sep, 2010-Feb, 2010-Apr, 2010-May

into four equally spaced intervals by time. Then for each academic year, the number of hosts for each time category and the number of hosts with incidents for each time category are counted. Then the proportion of incidents per category is divided by the proportion of hosts per category for each academic year to obtain the scaled incidence rates. Scaled incidence rates are calculated so that each academic year has equal weight. For some months, the ARP cache data was incomplete and network times could not be calculated for all categories. The months affected are shown in Table 7.3 and data from those months were not included in the plots shown in this section. However, when creating regression models in a later section, the maximum network time was substituted for the missing values.

Figure 7.9 shows boxplots of the scaled incidence rates for each time category for all hosts (left), "Housing" network hosts (middle), and "Other" network hosts (right). Hosts in the Q2 category appear in some cases to have a higher proportional incidence rate, but Q2 also shows a broader range in values than the other categories.

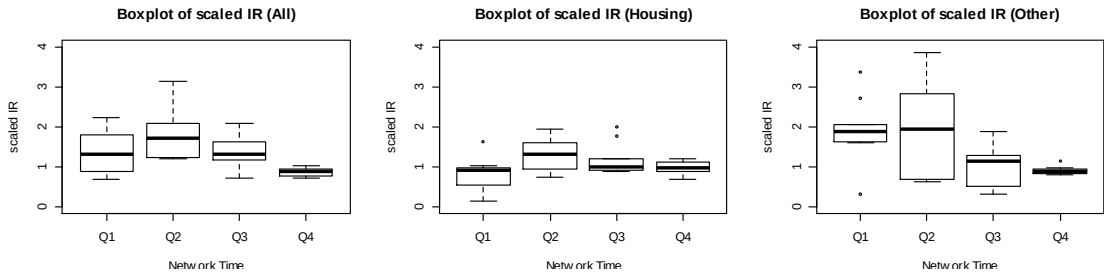

Figure 7.9: Boxplots of scaled incidence rates for network time categories.

7.2.7 On at beginning of month

Since the calculated time values do not differentiate between hosts that are always on and hosts that are on intermittently, an attempt to gauge if a machine is likely to be on all the time is made by looking at the time value of its first seen timestamp. A host was classified as "On" if its first seen timestamp is within 4 hours of the beginning of the month (between 00:00 and 04:00 of the first day), otherwise it was classified as "Off." The reason for the 4-hour window is due to how the ARP cache data is collected. The rationale is that hosts that are on at midnight at the beginning of the month are more likely to be on all the time when compared to hosts not on at midnight at the beginning of the month. Similar to network time, there is uncertainty that this value actually captures the information it is intended to measure.

Since this measurement also depends on the ARP cache timestamp information, the same months listed in Table 7.3 were removed from the analysis in this section. For creating regression models in a later section, the missing values were substituted with the value indicating the hosts were "On."

Figure 7.10 shows the scaled incidence rates for each group (On/Off) by

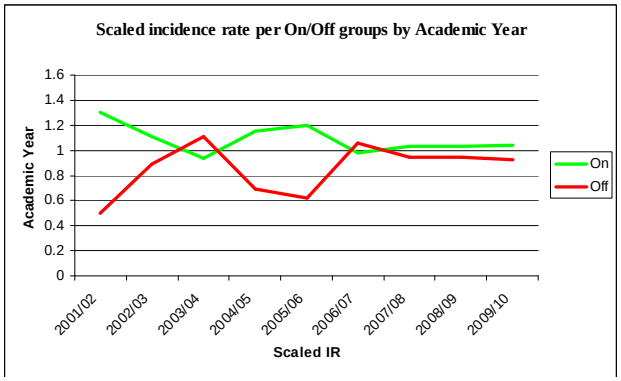

Figure 7.10: Scaled incidence rate per On/Off groups by Academic Year.

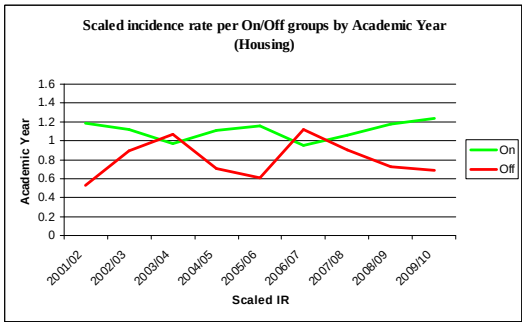

Figure 7.11: Scaled incidence rate per On/Off groups by Academic Year (Housing).

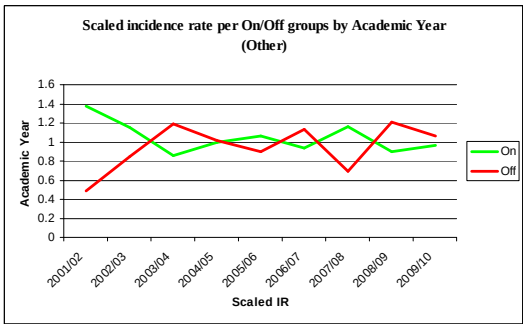

Figure 7.12: Scaled incidence rate per On/Off groups by Academic Year (Other).

academic year for the full population while Figure 7.11 shows the same information for the "Housing" network population and Figure 7.12 for the "Other" network population. For the last few academic years (2007/08, 2008/09, 2009/10), the relationship appears to be fairly consistent for the population as a whole with machines indicated as being "On" at the beginning of the month have a slightly higher proportional rate of incidents compared to machines indicated as being "Off" at the beginning of the month. For the same academic years, the difference between these two groups seems to be increasing for the "Housing" network population, while it has flipped back and forth for the "Other" network population.

7.3 Modeling and Evaluation Methodology

Logistic regression models were created using full sets of data from either one academic calendar year or from three consecutive academic calendar years. Logistic regression is appropriate for modeling events with binary dependent variables [94]. Univariate models were created as well as models containing the full set of variables. The area under the ROC curve (AUC) was calculated for each model to evaluate its classification ability. Due to the unknown number of false negatives (undetected incidents) in the data, some measures may be biased. This biasing situation also occurs with medical screening tests [95] where follow-up evaluations are not performed if an initial screening test is negative.

Models were also evaluated by using the logistic regression model outputs for each entry as a set of probability weights in combination with unequal probability sampling. Sample sizes were set based on the total number of host-months and incidents within a data set so there would be 10 expected incidents (true positives) within selected samples if all entries had an equal probability (the population incidence rate) of being involved in a computer security incident. The number of true positives was recorded for each sample and the sampling was repeated 100 times using the model data. The results were then averaged to obtain a measurement value. The expected number of incidents was kept constant to allow for comparisons across years even though the number of host-months and incidents varies for each time period. A value of 10 should be large enough to obtain a nontruncated distribution of results from working with nonnegative values. Table 7.4 summarizes the data

Table 7.4: Data and sizes of samples used for evaluations of models and forecasts.

Data used for models and/or forecasts	Total # of host-months	Total # of incidents	Incidence rate (per 1,000)	Size of samples
Single-year				
2001/02	301,033	375	1.25	8,028
2002/03	315,304	731	2.32	4,314
2003/04	318,812	4,348	13.64	734
2004/05	329,211	1,694	5.15	1,944
2005/06	337,087	1,011	3.00	3,335
2006/07	342,961	669	1.95	5,127
2007/08	309,485	282	0.91	10,975
2008/09	268,942	122	0.45	22,045
2009/10	246,142	102	0.41	24,132
Three-year				
2001/04	935,149	5,454	5.83	1,715
2002/05	963,327	6,773	7.03	1,423
2003/06	985,110	7,053	7.16	1,397
2004/07	1,009,259	3,374	3.34	2,992
2005/08	989,533	1,962	1.98	5,044
2006/09	921,388	1,073	1.16	8,587

sets and sample sizes used. Calculations were performed using the R statistical software [68]. An example of the R code used for the logistic regression models is provided in Appendix E.

The performance of the models on testing data (data not used for creation of the models) was evaluated using a similar process. Logistic regression outputs were calculated for the single academic year that follows the data set used for model construction. Model outputs from the testing data were used with unequal probability sampling to select hosts which may be involved in an incident. The sample sizes were selected so that there would be an expected number of 10 incidents in the sample if all entries had an equal probability (the population incidence rate) of being involved in an incident. Sampling was repeated 1,000 times and the results (number of true positives in the sample) were averaged to obtain a value that could be compared

across models.

Models were created using from one or three consecutive years of data to determine if including more historical data added any benefit to the forecasts. On one hand, since threats and defenses evolve over time, what occurred in the past may not be very relevant to identifying current or predicting future incidents. On the other hand, although technical aspects (such as the particular vulnerabilities exploited and the prevalence of defensive measures such as host-based firewalls and antivirus software) may change over time, some underlying aspects (such as user-related behaviors and motivations) may be more stable and remain relevant over time.

7.4 Modeling and Evaluation Results

Figure 7.13 shows the "area under the ROC curve" (AUC) values and the number of true positives in the sampling results for models created using a single year of data applied to both the model data and the testing data (the academic year immediately following the model data). A higher AUC value is often interpreted as indicating better discriminating ability of the model [96] and higher AUC values appear to be associated with higher numbers of true positives in the samples from the model data. Overall, models using all of the variables have higher AUC values than the univariate models and higher numbers of true positives for both the model and testing data samples. The "Housing/Other" variable seems to provide better performance than most other individual variables, especially for more recent years

(such as 2008/09), although the "Calendar.Month" variable also has higher AUC values compared to other individual variables for some of the years in the middle (such as 2003/04 and 2005/06).

For five out of the eight years of testing data shown in Figure 7.13, models using all of the variables had the highest number of true positives in samples, and were very close to the best for a sixth year. The model based on the "Housing/Other" variable had the highest number of true positives for the 2003/04 testing data compared to other models, and typically was the second best performer for other years. While the model based on "Age" had the highest number of true positives for the 2002/03 testing data, the improvement over assuming an equal probability of incidents (based on the population incident rate) is negligible.

Figure 7.14 shows the AUC values for models and the number true positives of sampling results for models created using three consecutive years of data applied to both the model data and the testing data (the single academic year immediately following the model data). Same as for the single-year models, models using all of the variables have higher AUC values than the univariate models and more true positives in the model data samples. However, the number of true positives in the model data samples for the three-year models tends to be lower than the number of true positives in the model data samples for the one-year models. The "Housing/Other" variable models have higher AUC values than most other individual variables for later years, while "Calendar.Month" models have higher AUC values for some of the earlier years compared to other univariate models. Similar to the single-year models, three-year models using all of the variables had the highest number of true positives

in the testing data samples. The models based on the "Housing/Other" variable had the next highest number of true positives for all years of testing data samples. Table 7.5 summarizes the highest average number of true positives from the testing data samples for both the one-year and three-year models. For years with testing data samples available using both models, the one-year models perform better in all cases except one (2007/08), but the difference between the one-year and three-year models for 2009/10 is minimal. In general, it appears the performance (measured by the number of true positives in the testing data samples) of single-year models using all of the variables is often the best or near-best. Among the univariate models, the "Housing/Other" variable performs better for more recent years for both the one- and three-year models.

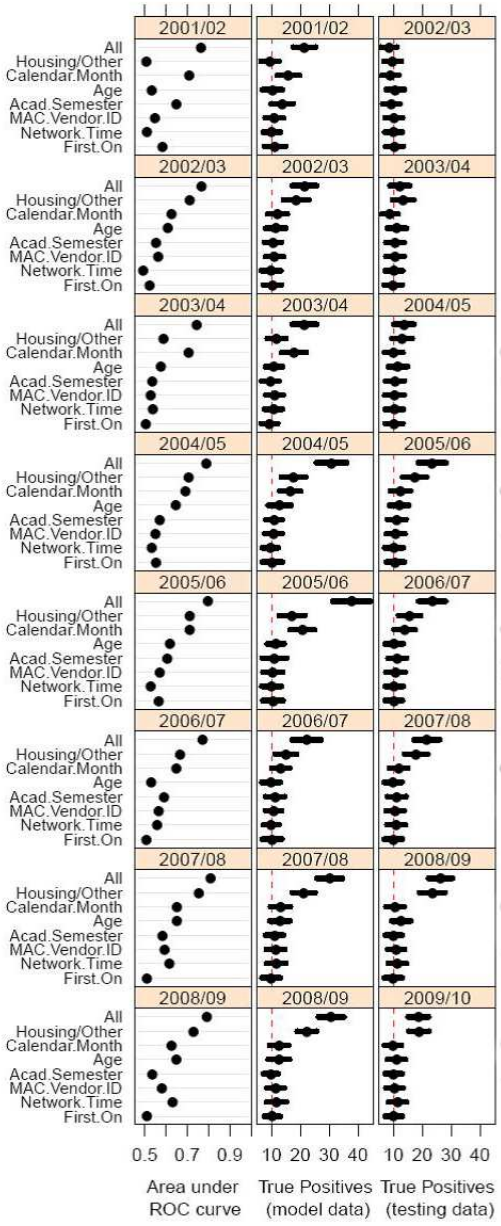

Figure 7.13: Summary of AUC values and number of true positives in samples for single-year models and testing data.

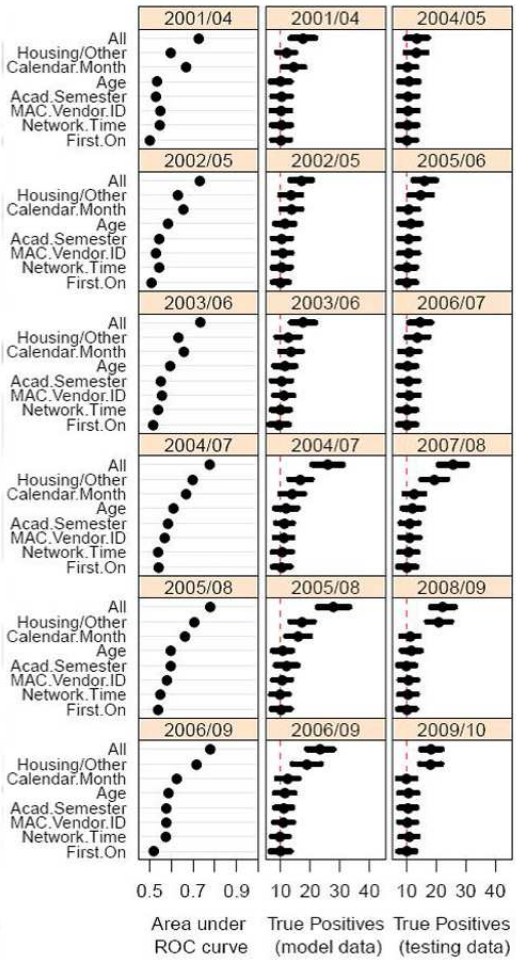

Figure 7.14: Summary of AUC vaues and number of true positives in samples for three-year models and testing data.

Table 7.5: Comparison of best forecasts.

	Single-year model			Three-year model		
Year of forecast data	Variables	mean	std.	Variables	mean	std.
2002/03	Age	10.6	3.2	-	-	-
2003/04	Housing/Other	13.3	3.7	-	-	-
2004/05	All	13.6	3.5	All	13.5	3.7
2005/06	All	23.4	4.8	All	16.1	3.9
2006/07	All	23.4	4.7	All	14.6	3.8
2007/08	All	21.5	4.4	All	25.6	4.8
2008/09	All	26.3	4.2	All	22.2	4.1
2009/10	Housing/Other	18.8	3.6	All	18.3	3.6

This chapter explored the association between different computer population characteristics and incident occurrence. The strongest association was seen based on whether a computer or device was part of the on-campus housing network or part of the remaining academic and research networks. Additional modeling could indicate if the effects of other factors would be observable for different individual incident types, or if targeted interventions based on population characteristic information would be beneficial.

Chapter 8: Conclusions and Future Work

This document explores the use of many types of models that can be applied to computer security incident data. Examples of how some of these models can be used with actual computer security incident data are also shown. Some of the observations and conclusions that can be made from the model illustrations and examples are enumerated and described below.

8.1 Observations and Conclusions

Although SIR and other types of infectious disease models have been applied to computer security incidents in the past, these applications typically have focused on the population as a whole and have proposed interventions that were dependent on this assumption. By using a stochastic SIR model, the effects on a subpopulation can be examined in more detail and intervention actions limited in scope to the subpopulation can be evaluated. In this scenario, the larger population acts as a reservoir of potential infection sources in addition to any infection sources already present in the subpopulation. Connections that cross the boundary between the larger population and the subpopulation can provide information regarding external threats and attack activity. If the larger population is sufficiently greater in size than the

subpopulation, observed incidents within the subpopulation are likely to be caused by external sources rather than other internal sources. This indicates that blocking attack activity at the border of a subpopulation could be an effective measure to reduce the number of internal infections. This kind of blocking intervention would be more effective than another type of intervention that depends on being applied to the overall population in its entirety.

Another assumption commonly included in deterministic SIR models that include a vaccination (or patching) parameter, is that the rate parameter for vaccination remains relatively constant over the time period of interest. As previously discussed, this assumption may not fit with the way patching activity actually takes place. Easier to patch and centrally managed computers are more likely to be patched first (which would more quickly reduce the number of susceptible computers), but then other computers may be more difficult to locate or undesirable to patch and may remain in a susceptible or vulnerable state. One type of stochastic SIR model that associates the rate of patching with the number or proportion of susceptible computers is presented and simulations indicate this type of model is more likely consistent with a set of incident data observed on a university network.

While the SIR models used for incident types that can spread directly from computer to computer without the need for specific user action usually assume a fully connected network topology (every node is directly connected to every other node), incident types that propagate through email usually involve less connected network topologies. Even when the same number of connection links is present and the average node degree is similar, different spreading behavior can result from

different arrangements or topologies. A scale-free or power-law network is a common topology used to describe social networks and simulations indicate this topology affects the number of observed incidents and is more likely consistent with email related incident data observed on a university network than a randomly connected topology.

Besides connection topology, email propagation can depend on user factors, such as the likelihoods of users to open infected messages. Some models have included this factor, but have treated all users as having the same likelihoods to open infected messages. A stochastic model has been presented where individual users are assigned a threshold value based on an exponential distribution and this threshold value is used to determine if a user opens infected messages. Simulations indicate this type of model (where most users are unlikely open infected messages, but a few are very likely to do so) is more likely consistent with observed email related incident data on a university network than a model that uses a uniform random distribution to assign threshold values. This indicates that accounting for and including user related factors in models can be important for matching observations and improving model performance. This can be important when considering types of interventions to use and evaluating their potential impact.

For example, if blocking or filtering of infected messages is applied, its impact may not be directly related to the overall number of infected messages blocked, but rather by the number of infected messages blocked which are being sent to a subset of nodes which are highly connected and have associated users more likely to open infected messages. Some form of targeted intervention (such as patching or

user awareness training) for this subset could produce better results in reducing the spread of infections than a more uniform application of intervention measures.

Targeted interventions may also be more effective for reducing the number of other computer security incident types even when infectious disease models are not applicable. Identifying and exploring a risk factor with a model (such as the likelihood a user will open an infected message) can be a simpler task than isolating and measuring such a risk factor in an actual setting. However, other observable or quantifiable factors may correlate or be associated with a risk factor and provide an opportunity for guiding intervention efforts. Observed incidents can be compared with observable characteristics of a population to try to identify potential factors that correlate or contribute to the risk of incident occurrence.

Modeling can be used to get a better understanding of factors that affect the occurrence and spread of computer security incidents. Examples of some compartmental infectious disease models have been presented and compared with incident data. These comparisons show that models using a variable patching rate may be a better reflection of actual patching behavior. Also, for incidents propagating by email, a scale-free network topology provides a better match to observed incident data compared to when a random network topology is used. Incidents requiring some type of user action for propagation (such as in the presented email propagation models) are more appropriately modeled using an exponential based distribution rather than a uniform random based distribution for representing the likelihood particular users will perform an action.

To the extent that these types of models aptly describe the observed data, they

can be used to test and explore some common intervention options applicable to infectious diseases. However, some incident types are not adequately modeled or described as an infectious process and other interventions may be more applicable. Regardless of the type of intervention actions taken, it is useful to have a way to gauge or measure the impact of such actions. One way to do so is to compare forecasts of the number of expected incidents before and after an intervention was started. If the observed number of incidents is less than the number forecast, then this difference may be attributable to the intervention actions taken.

Software reliability growth models and time series models are two types of forecasting models than can be applied to a range of incident types. Obtaining estimates for model parameters from observed or collected data can be simpler and faster for these models than for some of the compartmental models described for modeling infectious diseases. The models themselves do not account for or include any intervention actions and therefore are not as useful for simulating interventions, however, they can be used for forecasting and then tracking and monitoring the impact of applied interventions. As illustrated with the "spamrelay" example in Chapter 6 using the Duane model with additional data, this process can also be used in reverse. Observed differences between model forecasts and actual observations can indicate when something influential has changed (either increasing or decreasing the number of expected observed incidents) and whether the selected forecasting model is still applicable or if it should be adjusted or replaced. Software reliability growth models appear better for forecasting incidents types attributable to known causes or vulnerabilities compared to time series models.

8.2 Future Work

The connection activity crossing or blocked at the border of subpopulation using an SIR model is in some ways analogous to the observed attack activity blocked or logged by an IDS/IPS that examines border traffic on a network. The modeled border activity in an SIR model could be compared with observed IDS/IPS detection alerts to see if external attack activity is consistent with expected activity from an infectious process, or if the attack activity is either more targeted or random in nature.

Similarly, the email propagation model includes information regarding the levels of infected messages filtered or blocked over time that could be compared to mail filter logs. This could provide additional information regarding aspects of the model (such as connection topology and distribution of user likelihoods for opening infected messages). The origins of infected or blocked messages could also be examined to determine if internal or external sources pose more of a risk.

Although the email propagation model includes some aspects related to the human users of the computers, none of the models include aspects related to human attackers who may be responsible for some of the observed attack activity. For example, is there any evidence of feedback from intervention efforts on relevant attack activity? If a certain type of attack becomes less effective because of intervention efforts, does attack decline as attack resources are applied elsewhere, or do occurrences of the attack activity increase to make up for the decline in successful attacks?

It is becoming more common for computer security incidents to be motivated in

part by financial incentives. A successful compromise or attack may financially benefit the attacker in some way. Identifying or classifying apparent motivations behind attacks could lead to better modeling and intervention planning as interventions that reduce attacker motivations for targeting resources could be more effective than interventions that simply thwart a particular attack venue and do not reduce attacker motivations.

Modeling can be a useful tool for gaining a better understanding of factors that can affect the occurrence of computer security incidents. A better understanding of these factors can lead to more effective use of limited resources available for intervention efforts. Modeling can also be a useful tool for evaluating the effectiveness of applied interventions and identifying if changes are needed to obtain the desired results.

Appendix A: Incident Data

A.1 Description

- Included below are the incident data for the "worm_msblast", "worm_nachi", "bagle_worm", "virus_klez", "virus_agobot", and "spamrelay" incident types.
- Each number represents an observed incident and the number itself represents the day the incident observation was recorded. Day 0 corresponds to June 13, 2001.

A.2 Worm_msblast data

790, 790,

790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790,
790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790,
790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790,
790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790,
790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790,
790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790,
790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790,
790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790,
790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790,
790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790,
790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790,
790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790,
790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790,
790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790,
790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790,
790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790,
790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790, 790,
790, 790, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791,
791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791,
791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791,
791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791,
791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791,
791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791,
791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791,
791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791,
791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791,
791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791,
791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 791, 792, 792, 792, 792,
792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792,
792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792,
792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792,
792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792,
792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792,
792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792,
792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792,
792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792,
792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792, 792,
792, 792, 792, 792, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793,
793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793,
793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793,
793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793,
793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793,
793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793, 793,
793, 793, 794, 794, 794, 794, 794, 794, 794, 794, 794, 794, 794, 794, 794, 794, 794,
794, 794, 794, 794, 794, 794, 794, 795, 795, 795, 795, 795, 795, 795, 795, 795, 795,

795, 795, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796,
796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796,
796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796,
796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796,
796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796,
796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796,
796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796,
796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796, 796,
796, 796, 796, 796, 796, 796, 796, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797, 797,
797, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798,
798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798,
798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798,
798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798,
798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798, 798,

798, 799, 799, 799, 799, 799, 799, 799, 799, 799, 799, 799, 799, 799, 799, 799, 799, 800, 800, 800, 800, 800, 800, 800, 800, 800, 800, 800, 801, 801, 801, 801, 801, 801, 801, 801, 801, 801, 801, 801, 801, 801, 801, 801, 801, 801, 801, 801, 801, 804, 804, 804, 804, 804, 804, 804, 804, 804, 804, 804, 804, 804, 804, 804, 804, 804, 804, 804, 806, 806, 806, 806, 810, 811, 811, 811, 811, 811, 812, 812, 812, 812, 812, 812, 812, 812, 812, 813, 813, 813, 813, 813, 813, 813, 817, 817, 817, 817, 817, 817, 817, 817, 817, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 818, 819, 819, 819, 819, 819, 819, 819, 819, 819, 819, 819, 819, 819, 819, 819, 819, 819, 819, 819, 820, 820, 820, 820, 820, 820, 820, 820, 820, 820, 821, 821, 821, 821, 821, 821, 821, 821

A.3 Worm_nachi data

825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 825, 826, 826, 826, 826, 826, 826, 826, 826, 826, 826, 826, 826, 826, 826, 826, 826, 826, 826, 826, 826, 826, 826, 826, 826, 826, 826, 826, 826, 826, 826, 826, 831, 831, 838, 838, 838, 838, 838, 838, 838, 838, 838, 838, 838, 838, 838, 838, 838, 838, 838, 838, 838, 838, 838, 838, 838, 838, 838, 839, 839, 839, 839, 839, 839, 839, 839, 839, 839, 839, 839, 839, 839, 839, 839, 839, 839, 839, 839, 839, 839, 839, 839, 839, 839, 839, 839, 839, 839, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 840, 841, 841, 841, 841, 841, 841, 841, 841, 841, 841, 841, 841, 841, 841, 841, 841, 841, 841, 841, 848, 848, 848, 848, 848, 848, 848, 848, 848, 848, 848, 848, 848, 848, 848, 848, 848, 848, 848, 848, 848, 848, 848, 848, 849, 849, 849, 849, 849, 849, 849, 849, 849, 849, 849, 849, 852, 852, 852, 852, 853, 853, 853, 853, 853, 853, 853, 853, 853, 853, 854, 854, 854, 854, 854, 862, 863, 869, 869, 869, 869, 869, 869, 881, 881, 881, 881, 881, 888, 889, 889, 889, 889, 889, 889, 889, 889, 889, 889, 889, 889, 889, 889, 889, 889, 889, 889, 889, 890, 902, 903, 916, 916, 916, 917, 919, 971, 999, 1001, 1017, 1020, 1020, 1020, 1020, 1023

A.4 Bagle_worm data

951, 951, 951, 953, 953, 956, 956, 956, 956, 956, 956, 956, 956, 957, 957, 957, 957, 957, 957, 957, 957, 979, 979, 979, 979, 979, 979, 979, 979, 979, 979, 979, 979, 979, 979, 979, 979, 979, 979, 980, 980, 981, 981, 993, 993, 993, 994, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 995, 996, 996, 996, 996, 996, 996, 996, 996, 996, 996, 996, 996, 996, 996, 997, 997, 997, 997, 1001, 1001, 1001, 1001, 1001, 1001, 1001, 1001, 1001, 1001, 1001, 1001, 1002, 1002, 1003, 1003, 1006, 1007, 1007, 1007, 1007, 1007, 1015, 1015, 1016, 1019, 1020, 1020, 1020, 1020, 1020, 1020, 1021, 1021, 1021, 1022, 1022, 1023, 1028, 1036, 1036, 1037, 1041, 1048, 1048, 1048, 1048, 1049, 1049, 1049, 1049, 1050, 1050, 1050, 1050, 1050, 1050, 1050, 1050, 1050, 1050, 1050, 1050, 1050, 1050, 1050, 1050, 1051, 1051, 1051, 1051, 1051, 1052, 1055, 1055, 1059, 1063, 1077, 1077, 1129, 1129, 1129, 1129, 1132, 1132, 1132, 1132, 1132, 1132, 1132, 1132, 1132,

1132, 1132, 1132, 1132, 1132, 1133, 1134, 1135, 1135, 1139, 1139, 1141, 1154, 1154, 1169, 1171, 1172, 1172, 1173, 1174, 1177, 1188, 1197, 1203, 1204, 1204, 1204, 1204, 1204, 1204, 1204, 1204, 1204, 1204, 1205, 1205, 1209, 1210, 1227, 1227, 1234, 1237, 1237, 1237, 1237, 1237, 1237, 1237, 1237, 1237, 1237, 1237, 1238, 1238, 1238, 1238, 1238, 1241, 1244, 1246, 1247, 1273, 1325, 1325, 1344, 1560, 1560, 1562, 1562, 1562, 1562, 1562, 1562, 1563, 1563, 1566, 1566, 1567, 1568, 1568, 1568, 1570, 1575, 1616, 1616, 1629

A.5 Virus_klez data

244, 246, 246, 246, 246, 246, 246, 246, 246, 246, 246, 246, 246, 246, 246, 246, 246, 246, 246, 246, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 247, 248, 248, 248, 248, 248, 248, 248, 248, 248, 248, 248, 248, 248, 248, 248, 248, 248, 248, 248, 249, 249, 249, 249, 249, 249, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 250, 251, 251, 251, 251, 251, 251, 251, 251, 251, 251, 251, 251, 251, 251, 251, 251, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 252, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 253, 254, 254, 254, 254, 254, 254, 254, 254, 254, 254, 254, 254, 254, 254, 254, 256, 256, 256, 256, 256, 256, 256, 256, 256, 256, 256, 256, 256, 256, 256, 256, 256, 256, 256, 256, 256, 256, 257, 257, 257, 257, 257, 257, 258, 258, 258, 258, 258, 258, 258, 258, 258, 258, 258, 258, 258, 258, 258, 258, 258, 258, 258, 258, 258, 258, 258, 258, 258, 258, 258, 258, 258, 259, 259, 259, 259, 259, 259, 259, 259, 259, 259, 259, 259, 259, 259, 259, 260, 260, 261, 261, 261, 261, 261, 261, 261, 261, 261, 261, 261, 261, 261, 261, 261, 262, 262, 262, 262, 262, 262, 262, 262, 262, 262, 262, 263, 263, 264, 264, 264, 265, 265, 265, 265, 265, 265, 265, 265, 265, 265, 265, 265, 265, 267, 268, 268, 268, 271, 271, 271, 271, 271, 272, 272, 273, 273, 277, 277, 277, 277, 277, 277, 278, 279, 279, 279, 279, 279, 281, 281, 281, 281, 281, 281, 282, 287, 292, 292, 292, 292, 292, 292, 293, 293, 293, 294, 294, 295, 295, 295, 295, 296, 296, 299, 299, 299, 299, 299, 299, 299, 300, 300, 300, 300, 300, 300, 301, 301, 301, 302, 302, 302, 302, 302, 303, 303, 303, 303, 305, 305, 305, 305, 305, 305, 305, 305, 305, 306, 306, 306, 306, 308, 308, 308, 308, 309, 309, 309, 309, 309, 309, 309, 309, 309, 309, 310, 311, 311,

311, 313, 313, 313, 313, 313, 313, 313, 313, 314, 314, 314, 314, 314, 314, 316, 316, 316, 316, 316, 316, 317, 317, 317, 317, 317, 317, 317, 317, 320, 320, 320, 320, 320, 320, 320, 320, 320, 320, 320, 320, 320, 320, 320, 320, 321, 321, 321, 321, 322, 322, 322, 322, 322, 323, 323, 323, 323, 323, 323, 323, 323, 323, 323, 323, 323, 324, 324, 324, 325, 325, 325, 325, 325, 326, 326, 326, 326, 327, 327, 327, 327, 328, 328, 328, 328, 328, 328, 328, 328, 328, 328, 328, 328, 328, 328, 328, 329, 329, 329, 329, 329, 329, 329, 329, 329, 329, 329, 329, 329, 329, 329, 329, 329, 329, 330, 331, 331, 331, 331, 331, 334, 334, 334, 334, 334, 334, 334, 334, 334, 334, 334, 334, 334, 334, 334, 334, 335, 335, 337, 337, 337, 337, 337, 337, 337, 337, 337, 337, 337, 338, 338, 338, 338, 338, 338, 338, 341, 341, 341, 341, 341, 341, 341, 341, 342, 342, 342, 342, 342, 342, 342, 343, 343, 349, 349, 350, 352, 358, 426, 439, 440, 443, 443, 443, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 447, 448, 448, 448, 448, 449, 449, 449, 450, 453, 453, 453, 453, 453, 453, 453, 454, 455, 455, 455, 456, 456, 456, 457, 457, 457, 457, 461, 461, 461, 461, 461, 461, 463, 463, 463, 463, 463, 463, 463, 463, 463, 463, 463, 463, 463, 463, 463, 464, 464, 464, 468, 468, 468, 468, 468, 473, 473, 474, 474, 474, 475, 475, 477, 477, 477, 477, 477, 481, 481, 481, 481, 481, 481, 481, 481, 483, 483, 483, 483, 483, 485, 485, 488, 488, 488, 488, 488, 488, 490, 490, 490, 490, 490, 492, 492, 495, 495, 495, 497, 497, 497, 497, 497, 497, 497, 499, 499, 499, 499, 502, 502, 502, 502, 502, 502, 506, 506, 506, 506, 506, 506, 506, 509, 509, 509, 509, 509, 512, 512, 512, 512, 512, 512, 512, 512, 516, 516, 516, 516, 516, 517, 517, 517, 517, 517, 517, 518, 518, 518, 518, 519, 519, 520, 520, 520, 520, 523, 523, 523, 523, 523, 527, 527, 527, 527, 527, 527, 527, 527, 531, 531, 531, 531, 531, 531, 540, 540, 544, 544, 544, 544, 544, 545, 545, 545, 546, 546, 546, 547, 547, 547, 547, 548, 548, 548, 548, 548, 548, 551, 551, 551, 551, 551, 551, 573, 573, 579, 579, 581, 582, 594, 594, 594, 594, 594, 594, 595, 596, 596, 596, 596, 596, 597, 597, 597, 597, 597, 597, 600, 600, 600, 600, 600, 600, 601, 601, 601, 601, 601, 602, 602, 602, 602, 602, 608, 608, 608, 608, 608, 610, 611, 611, 611, 611, 611, 611, 611, 617, 617, 617, 617, 617, 617, 617, 617, 617, 618, 618, 618, 618, 621, 621, 621, 621, 621, 621, 621, 621, 621, 621, 621, 621, 621, 622, 622, 622, 622, 622, 623, 623, 623, 623, 624, 628, 628, 628, 628, 628, 629, 631, 631, 631, 631, 631, 632, 636, 636, 636, 636, 636, 636, 636, 637, 638, 638, 638, 638, 639, 642, 642, 642, 644, 645, 646, 656, 656, 656, 656, 656, 656, 658, 658, 659, 659, 659, 660, 660, 660, 663, 665, 670, 670, 670, 670, 671, 671, 674, 674, 677, 677, 677, 677, 677, 677, 677, 678, 678, 679, 681, 685, 686, 686, 686, 687, 687, 688, 692, 692, 693, 693, 694, 694, 694, 694, 698, 698, 698, 699, 700, 700, 705, 705, 705, 705, 705, 706, 706, 708, 756, 826, 848, 852, 859, 902

A.6 Virus_agobot data

852, 852, 852, 852, 853, 853, 853, 854, 854, 854, 854, 854, 854, 855, 855, 855, 861, 861, 861, 861, 861, 862, 862, 862, 866, 866, 866, 867, 867, 867, 867, 985, 1008, 1008, 1008, 1008, 1008, 1008, 1009, 1010, 1013, 1013, 1015, 1015, 1016, 1017, 1021,

1021, 1022, 1022, 1022, 1022, 1022, 1022, 1022, 1022, 1022, 1022, 1022, 1022, 1024, 1027, 1028, 1029, 1029, 1029, 1029, 1029, 1029, 1029, 1029, 1029, 1029, 1029, 1030, 1031, 1031, 1031, 1031, 1031, 1031, 1031, 1031, 1031, 1031, 1031, 1031, 1033, 1033, 1033, 1033, 1033, 1033, 1033, 1034, 1035, 1035, 1035, 1035, 1035, 1035, 1035, 1035, 1035, 1035, 1035, 1036, 1036, 1036, 1036, 1036, 1036, 1036, 1036, 1036, 1036, 1036, 1036, 1036, 1036, 1036, 1036, 1036, 1036, 1036, 1037, 1037, 1037, 1037, 1037, 1038, 1038, 1038, 1039, 1039, 1040, 1040, 1040, 1040, 1040, 1041, 1041, 1042, 1042, 1042, 1042, 1042, 1042, 1042, 1042, 1042, 1042, 1043, 1043, 1043, 1043, 1044, 1044, 1044, 1044, 1044, 1044, 1044, 1044, 1044, 1044, 1045, 1045, 1045, 1045, 1045, 1045, 1045, 1045, 1045, 1045, 1045, 1045, 1045, 1045, 1045, 1045, 1046, 1047, 1048, 1048, 1048, 1048, 1048, 1048, 1048, 1048, 1048, 1048, 1048, 1048, 1048, 1048, 1049, 1049, 1049, 1050, 1050, 1050, 1050, 1051, 1052, 1052, 1052, 1052, 1052, 1052, 1052, 1054, 1054, 1054, 1054, 1054, 1054, 1054, 1054, 1054, 1054, 1055, 1056, 1056, 1056, 1056, 1056, 1056, 1056, 1056, 1056, 1056, 1056, 1057, 1057, 1059, 1062, 1063, 1064, 1064, 1064, 1064, 1064, 1064, 1076, 1077, 1080, 1080, 1080, 1084, 1084, 1086, 1087, 1088, 1090, 1090, 1091, 1094, 1097, 1121, 1121, 1121, 1121, 1121, 1121, 1121, 1121, 1121, 1121, 1203, 1231, 1553

A.7 Spamrelay data

A.7.1 Original time interval

363, 552, 703, 728, 740, 755, 764, 768, 768, 775, 778, 791, 799, 800, 802, 803, 803, 806, 807, 807, 817, 818, 820, 822, 824, 824, 824, 824, 831, 831, 831, 831, 831, 831, 831, 838, 838, 838, 838, 838, 838, 839, 839, 839, 839, 839, 840, 840, 840, 840,

840, 840, 841, 841, 842, 842, 842, 847, 853, 853, 859, 911, 928, 966, 966, 971, 975, 978, 1000, 1020, 1021, 1023, 1023, 1027, 1028, 1030, 1034, 1034, 1034, 1040, 1042, 1044, 1047, 1048, 1049, 1054, 1055, 1057, 1057, 1057, 1059, 1059, 1063, 1064, 1066, 1076, 1090, 1093, 1103, 1121, 1127, 1153, 1171, 1172, 1172, 1172, 1174, 1174, 1175, 1175, 1175, 1175, 1175, 1175, 1181, 1184, 1199, 1203, 1208, 1216, 1275, 1280, 1312, 1312, 1312, 1313, 1313, 1321, 1323, 1325, 1328, 1337, 1339, 1339, 1353, 1359, 1366, 1366, 1366, 1366, 1366, 1366, 1385, 1385, 1385, 1409, 1409, 1414, 1415, 1419, 1420, 1420, 1420, 1421, 1421, 1421, 1422, 1423, 1423, 1434, 1434, 1434, 1434, 1434, 1434, 1434, 1434, 1434, 1434, 1434, 1434, 1435, 1435, 1435, 1435, 1435, 1436, 1436, 1448, 1450, 1450, 1454, 1462, 1476, 1476, 1478, 1479, 1489, 1496, 1496, 1496, 1514, 1531, 1532, 1536, 1546, 1556, 1556, 1556, 1556, 1556, 1556, 1556, 1556, 1556, 1557, 1557, 1558, 1558, 1558, 1562, 1565, 1566, 1567, 1567, 1567, 1567, 1568, 1568, 1570, 1570, 1570, 1573, 1573, 1573, 1575, 1575, 1580, 1582, 1582, 1587, 1589, 1590, 1590, 1590, 1591, 1596, 1596, 1602, 1602, 1602, 1609, 1611, 1611, 1611, 1612, 1616, 1617, 1618, 1619, 1622, 1628, 1629, 1629, 1629, 1629, 1631, 1636, 1636, 1637, 1638, 1639, 1640, 1640, 1646, 1648, 1649, 1653, 1655, 1671, 1672, 1673, 1685, 1685, 1686, 1686, 1686, 1687, 1687, 1688, 1689, 1692, 1692, 1692, 1700, 1702, 1705, 1708, 1710, 1710, 1713, 1714, 1714, 1714, 1714, 1715, 1720, 1720, 1721, 1721, 1721, 1723, 1735, 1735, 1735, 1735, 1735, 1736, 1748, 1762, 1762, 1769, 1769, 1769, 1786, 1786, 1786, 1786, 1786, 1786, 1786, 1786, 1786, 1819, 1829, 1834, 1836, 1855, 1867, 1883, 1889, 1894, 1896, 1902, 1904, 1904, 1904, 1904, 1904, 1904, 1904, 1904, 1904, 1906, 1916, 1916, 1916, 1916, 1916, 1916, 1918, 1918, 1918, 1918, 1918, 1924, 1924, 1924, 1924, 1924, 1924, 1924, 1926, 1926, 1926, 1926, 1927, 1927, 1937, 1938, 1944, 1944, 1947, 1948, 1951, 1951, 1951, 1951, 1951, 1952, 1953, 1954, 1955, 1958, 1959, 1960, 1960, 1968, 1972, 1972, 1982, 1982, 1983, 1983, 1983, 1983, 1983, 1993, 1993, 1995, 1995, 1997, 2000, 2000, 2004, 2007, 2007, 2010, 2017, 2037, 2038, 2039, 2045, 2051, 2052, 2059, 2059, 2063, 2065, 2066, 2070, 2071, 2071, 2073, 2073, 2092, 2092, 2092, 2092, 2092, 2095

A.7.2 Additional time interval

2101, 2113, 2115, 2115, 2115, 2137, 2148, 2154, 2154, 2161, 2205, 2205, 2206, 2248, 2284, 2284, 2302, 2302, 2302, 2302, 2311, 2323, 2324, 2360, 2378, 2390, 2430, 2430, 2430, 2430, 2430, 2430, 2430, 2431, 2431, 2435, 2438, 2541, 2554, 2651, 2658, 2658, 2660, 2670, 2711, 2721, 2721, 2738, 2826, 2830, 2830, 2830, 2917, 2940, 2941, 2973, 3000, 3010, 3030, 3033, 3050, 3050, 3050, 3106, 3106, 3106, 3106, 3232

Appendix B: R code for SIR models

B.1 Description

Example of R code used for SIR models.

B.2 R code example for SIR models

```
# R Script to do simulate stochastic SIR models
########################################

### PARAMETER LIST ###
modelType <- c("A")    # STANDARD VACCINATION
seedVal <- 2014062611
Npop <- 1000           # Total population size
betaVal <- 1.0
gammaVal <- 0.5
alphaVal <- 0.025
TotalTVal <- 30
nReps <- 5

# Create fully connected adjacency matrix
Nvec <- rep(1,(Npop*Npop))
NmatA <- matrix(Nvec, nrow = Npop, ncol = Npop)
NmatID <- diag(Npop)
NmatB <- (NmatA - NmatID)

# Specify a random seed value
set.seed(seedVal)
seedList <- runif(Npop, 0, 999999)

#####################################################################
####### BELOW TO GENERATE SIR EXAMPLE WITH VACCINATION ##############
#####################################################################

####### START SIMULATION REPETITIONS #######

for (i in 1:nReps) {

seedVal2 <- seedList[i]
set.seed(seedVal2)

# Specify a starting vector of susceptibles (1 = susc., 0 = not susc.)
Svec <- rep(1, Npop)

# Specify a starting vector of infected nodes (1 = infected, 0 = not infected)
Ivec <- rep(0, Npop)

# Specify a starting vector of recovered nodes (1 = recovered, 0 = not recov.)
Rvec <- rep(0, Npop)

# Add a state vector for vaccinated nodes (S -> R transitions)
```

```
VacRvec <- Rvec

# Specify state vectors --initial infected node and remove from it from susceptible
    list (and subpop list)
Ivec[1] <- 1
Svec[1] <- 0


# Specify recovery rate--gamma
gamma <- gammaVal

# Specify infection rate--beta (can vary for external by setting a "ro" value)
beta <- betaVal

# Specify vaccination rate--alpha (alpha2) for internal network
alpha <- alphaVal

# Average degree of node; needed for S -> I transition check
# mvalue <- (Npop - 1)
mList <- rep(1, Npop)
mvalue <- mean(NmatB %*% mList)



# Specify total time and time steps and calculate number of iteration
##############################################
## Specify duration and storage for simulation
TotalT <- 30
# Specify time step--deltaT
deltaT <- 0.1
niter <- (TotalT/deltaT)
Starray <- matrix(,Npop,(niter+1))
Itarray <- matrix(,Npop,(niter+1))
Rtarray <- matrix(,Npop,(niter+1))
Ktarray <- matrix(,Npop,(niter+1))
VacRtarray <- matrix(,Npop, (niter+1))

Starray[,1] <- Svec
Itarray[,1] <- Ivec
Rtarray[,1] <- Rvec
Ktarray[,1] <- (NmatB %*% Ivec)
VacRtarray[,1] <- VacRvec

### Adding interator to check for transitions at each time step
 #
 #
###

for (j in 1:niter) {

#####################################################################
## Branch "B" code files reverse the order of transition checks  ##
## S -> I; then I -> R; then S -> R when vaccination added       ##
#####################################################################

#######################################################
## Check for transitions from Susceptible to Infected
# Somewhere calculate an "mvalue" (average degree of network), but just specify for
# now (or "Npop - 1" for fully connected)

# Calculate the number of infected neighbors (K) for each node
Kvec <- NmatB %*% Ivec

# Loop through and look for transitions
for (i in 1:Npop) {
  if (Svec[i] == 1) {
    randvalue <- runif(1,0,1)
    if (randvalue < ((beta/mvalue)*(Kvec[i])*deltaT)) {
```

```
      Ivec[i] <- 1
      Svec[i] <- 0
    }
  }
}

#####################################################
## Check for transitions from Infected to Recovered

# Add a delay for first recovered

if (j < (2.0/deltaT)) {
  gamComp <- 0
} else {
  gamComp <- gamma
}


# Loop through and look for transitions
for (i in 1:(Npop)) {
  if (Ivec[i] == 1) {
    if (runif(1,0,1) < (gamComp*deltaT)) {
      Ivec[i] <- 0
      Rvec[i] <- 1
    }
  }
}

#####################################################################
## Check for transitions from Susceptible to Recovered (vaccination)

# Loop through and look for transitions
for (i in 1:(Npop)) {
  if(Svec[i] == 1) {
    if (runif(1,0,1) < (alpha*deltaT)) {
      Svec[i] <- 0
      Rvec[i] <- 1
      VacRvec[i] <- 1
    }
  }
}

#################################################################
## After iteration, add Svec, Ivec, Rvec, Kvec to storage array

Starray[,(j+1)] <- Svec
Itarray[,(j+1)] <- Ivec
Rtarray[,(j+1)] <- Rvec
Ktarray[,(j+1)] <- Kvec
VacRtarray[,(j+1)] <- VacRvec

### First need to add an iteration loop to the transition checks,
# use "j" as iterator

### Closing iterator for transition checks based on total time
 #
 #
###
}

### SAVE SOME VALUES AND OUTPUTS TO A FILE ###

recoveredIValues <- (colSums(Rtarray) - colSums(VacRtarray))
recIValues <- recoveredIValues[seq(1, length(recoveredIValues), (1/deltaT))]
finalInfected <- tail(recoveredIValues, 1)
totalTimeInf <- sum(colSums(Itarray)*deltaT)
paramValues <- c(modelType, Npop, TotalTVal, niter, gammaVal, betaVal, alphaVal,
    seedVal2, finalInfected, totalTimeInf)
```

```
179
180 write(paramValues, ncolumns = 10, file = "poutputA.txt", append = TRUE)
181 write(recIValues, ncolumns = c(niter+1), file = "recIValuesA.txt", append = TRUE)
182
183
184 ###### END SIMULATION REPETITIONS #######
185 }
```

Codes/Standard–Vaccination–Example–03D.R

Appendix C: R code for Email models

C.1 Description

Example of R code used for Email models.

C.2 R code example for Email models

```
# R script to simulate stochastic SIR model for Email virus/worm propagation
################################################

# Load "igraph" package
library(igraph)

### PARAMETER LIST ###
modelType <- c("D")
seedVal <- 21406051
NpopMax <- 10000    # Total population size
ePowerVal <- 1    # Scale-free topology parameter
avgContactsVal <- 513 # Scale-free topology parameter [513 -> 0.1]
openVecDefault <- 0.10
expLH <- 0.10   # expected value exponential for user-likelihood for opening
    messages
gammaVal <- 0.3
betaVal <- 1.0
alphaVal <- 0.0
blockRate <- 0.0
blockDelay <- 0   # number of iterations to delay blocking
roVal <- (1 - blockRate)  # 1 - (block rate)
TotalTVal <- 90
nReps <- 1    # Number of simulation repetitions to run


## Specify a random seed value
set.seed(seedVal)
seedList <- runif(NpopMax,0,999999)

####### START SIMULATION REPETITIONS #######

for (i in 1:nReps) {

iterCount <- i

seedVal2 <- seedList[i]
set.seed(seedVal2)

#### SET DYNAMIC PARAMETER VALUES ####
gammaVal <- runif(1, min=0.2, max=1.0)
betaVal <- runif(1, min=0.2, max=1.2)
alphaVal <- runif(1, min=0.005, max=0.2)
avgContactsVal <- runif(1, min=513, max=1056)
expLH <- runif(1, min=0.05, max=0.15)
```

```
Npop <- 10000

### SCALE-FREE GRAPH using "igraph" package functions
ePower <- ePowerVal
avgContacts <- avgContactsVal
NmatA <- barabasi.game(Npop, power=ePower, m=avgContacts, out.pref=TRUE,directed=
    FALSE)
NmatB <- get.adjacency(NmatA, sparse=FALSE)

## Specify initial starting states of MACHINES (S, I, R)

# Susceptibles (1 = susc., 0 = not susc.)
Svec <- rep(1, Npop)

# Infecteds (1 = infected, 0 = not infected)
Ivec <- rep(0, Npop)  # Infected nodes

# Recovereds (1 = recovered, 0 = not recovered)
Rvec <- rep(0, Npop)


## Specify number of infected messages (per node)
KNvec <- rep(0, Npop)  # New infected messages
KOvec <- rep(0, Npop)  # Previously unopened infected messages
KTvec <- rep(0, Npop)  # Total infected messages

## Storage vector for infection transition value check
iValueVec <- rep(0, Npop) # Store value that is compared to random number for
    transition check

## Specify a node (associated user) likelihood to open infected emails
#### Set low value default (except when testing "always" open scenario)
#OpenVec <- rep(0.001, Npop)

OpenVec <- rep(openVecDefault, Npop)

#### For user LH based on a uniform random distribution
#for (i in 1:Npop) {
#    OpenVec[i] <- (runif(1,0,(2*openVecDefault)))
#}

#### For user LH based on an exponential distribution
for (i in 1:Npop) {
    OpenVec[i] <- (1-pexp(runif(1,0,1),(1/expLH)))
}

## Specify initial infection start--first node transitions from Susceptible to
    Infected
Ivec[1] <- 1
Svec[1] <- 0


## Specify some generic worm/virus parameters

# Recovery rate--gamma
gamma <- gammaVal

# Infection rate--beta
beta <- betaVal

# Patching rate--alpha
alpha <- alphaVal

# Mail server passing rate (1 means all pass, no blocking; 0 means all blocked/
    deleted)
ro <- roVal


```

```
## Specify total time and time steps and calculate number of iterations
TotalT <- TotalTVal
numTSteps <- 1
deltaT <- (1/numTSteps)
niter <- (TotalT/deltaT)

# Define some parameter rates, distributions, etc.
infMessRate <- 0.1*deltaT

#### Create Counter for I -> R transitions ####
countIR <- 0
counterIR <- rep(0, (niter+1))

## Create storage Arrays
Starray <- matrix(,Npop,(niter+1))
Itarray <- matrix(,Npop,(niter+1))
Rtarray <- matrix(,Npop,(niter+1))
KNtarray <- matrix(,Npop,(niter+1))
KOtarray <- matrix(,Npop,(niter+1))
KTtarray <- matrix(,Npop,(niter+1))
CVtarray <- matrix(,Npop,(niter+1))
iValueArray <- matrix(,Npop,(niter+1))

## Initialize storage Arrays for t=0
Starray[,1] <- Svec
Itarray[,1] <- Ivec
Rtarray[,1] <- Rvec
KNtarray[,1] <- (NmatB %*% Ivec)
KOtarray[,1] <- KOvec
KTtarray[,1] <- KNtarray[,1] + KOtarray[,1]
iValueArray[,1] <- iValueVec

## Go ahead and create full array for email checking
# For Uniform Random Checking
for (i in 1:Npop) {
  for (j in 1:(niter+1)) {
    CVtarray[i,j] <- round(runif(1,0,1))
  }
}

## TESTING ALTERNATE TIMINGS FOR EMAIL CHECKING
## binomial distribution where each node has different binomial probability
## that is exponentially distributed
#CVXtarray <- matrix(,100,100)
#binProb <- rep(0.5, 100)
#for (i in 1:100) {
#  binProb[i] <- pexp(runif(1,0,1),2)
#}
#
#for (i in 1:100) {
#  for (j in 1:100) {
#    CVXtarray[i,j] <- rbinom(1,1,(1-binProb[i]))
#  }
#}
#hist(rowSums(CVXtarray)

###
 #
 # Adding iterator to check and perform transitions at each time step
 #
###

for (j in 1:niter) {

#######################################################
## Proposed order of transition checks               ##
## I -> R; then S -> I; then S -> R when applicable ##
#######################################################
```

```
176
177 ## Delay I -> R transitions for first few iterations
178
179 if (j < (2.0)) {
180   gamComp <- 0
181 } else {
182   gamComp <- gamma
183 }
184
185 #####
186 ## Check I -> R transitions
187 for (i in 1:Npop) {
188   if (Ivec[i] == 1) {
189     if (runif(1,0,1) < (gamComp*deltaT)) {
190       Ivec[i] <- 0
191       Rvec[i] <- 1
192       countIR <- countIR+1
193     }
194   }
195 }
196
197
198 #####
199 ## Check S -> I transitions
200
201 # Calculate number of infected neighbors for susceptibles
202 nIvec <- (NmatB %*% Ivec)
203
204 # Calculate number of newly received infected messages for susceptibles
205 for (i in 1:Npop) {
206   KNvec[i] <- rbinom(1, nIvec[i], infMessRate)
207 }
208
209 ## If mail server not blocking (i.e. if "ro" = 1), add new infected messages
210 ## to existing infected messages
211
212 # Add a delay for when blocking is implemented
213
214 if (j < ((blockDelay+1)/deltaT)) {
215   roComp <- 1.0
216 } else {
217   roComp <- ro
218 }
219
220
221 for (i in 1:Npop) {
222   KTvec[i] <- KOvec[i] + (ro * KNvec[i])
223 }
224
225 # If node is susceptible and user checked messages, did user open any
226 # infected messages?
227
228 for (i in 1:Npop) {
229   if ((Svec[i] == 1) & (CVtarray[i,j] == 1) & (KTvec[i] > 0)) {
230     KOvec[i] <- 0
231     for (k in 1:(KTvec[i])) {
232       randvalue <- runif(1,0,1)
233       iValue <- (beta*OpenVec[i])
234       iValueVec[i] <- iValue
235       if (randvalue < iValue) {
236         Ivec[i] <- 1
237         Svec[i] <- 0
238       }
239     }
240   }
241 }
242
243 #####
```

```
## Check S -> R transitions
for (i in 1:Npop) {
  if (Svec[i] == 1) {
    if (runif(1,0,1) < (alpha*deltaT)) {
      Svec[i] <- 0
      Rvec[i] <- 1
    }
  }
}



########################################
## After iteration, add KNvec, KTvec, KOvec, Svec, Ivec, Rvec to storage array
Starray[,(j+1)] <- Svec
Itarray[,(j+1)] <- Ivec
Rtarray[,(j+1)] <- Rvec
KNtarray[,(j+1)] <- KNvec
KOtarray[,(j+1)] <- KOvec
KTtarray[,(j+1)] <- KTvec
iValueArray[,(j+1)] <- iValueVec

counterIR[(j+1)] <- countIR

###
 #
 # Closing iterator to check and perform transitions at each time step
 #
###

}

### SAVE SOME VALUES AND OUTPUTS TO A FILE ###
linkCount <- ecount(NmatA)
finalRecovered <- tail(colSums(Rtarray),1)
totalTimeInf <- sum(colSums(Itarray)*deltaT)
recoveredValues <- colSums(Rtarray)
finalIR <- tail(counterIR, 1)

paramValues <- c(iterCount, modelType, Npop, TotalTVal, ePowerVal, avgContactsVal,
    linkCount, gammaVal, betaVal, alphaVal, expLH, seedVal2, finalIR,
    finalRecovered, totalTimeInf)

write(paramValues, ncolumns=15, file="poutputEX4_S1.txt", append=TRUE)
write(recoveredValues, ncolumns=c(TotalT+1), file="recValuesEX4_S1.txt", append=
    TRUE)
write(counterIR, ncolumns=c(TotalT+1), file="IRtransValues_EX4_S1.txt", append=TRUE
    )

###### END SIMULATION REPETITIONS #######
}
```

Codes/4EX–BaselineD.R

Appendix D: R code for Time series models

D.1 Description

Example of R code used for Time series models.

D.2 R code example for Time series models

```
# ARMA/ARIMA models for UMD Incident Data

# Read in original (weekly) data from text file, difference it, subtract
# mean, and convert to time-series object.  Also, create cumulative values

orig.data=scan("data.txt")
orig.data <- c(0, orig.data)
data0 <- ts(cumsum(orig.data), start=1)
nobs <- length(data0)
plim <- 3
qlim <- 3
season <- 52
sdiff <- 0

# ADDED FOR SQUARE-ROOT TRANSFORMATION
# ip = inverse power, 2 means sqrt, 3 means cube root, etc.
ip = 2

data <- (data0)^(1/ip)

# data <- diff(orig.data)
# data.mean <- mean(data)
# data <- data - data.mean
# data=ts(data, start=1)
# data
# cumul.orig.data <- cumsum(orig.data)

# Create matrices for desired output values
# aics.r = AIC values reported by R function "arima"
# aics.m = AIC values calculated according to equation in Shumway(2000) and defined
     by his "sarima.R" function
# aicc.m = AICc values calculated according to Shumway's "sarima.R" function
# bics.m = BIC values calculated according to Shumway's "sarima.R" function

aics.r <- matrix(, 1+plim, 1+qlim, dimnames=list(p=0:plim, q=0:qlim))
aics.m <- matrix(, 1+plim, 1+qlim, dimnames=list(p=0:plim, q=0:qlim))
aicc.m <- matrix(, 1+plim, 1+qlim, dimnames=list(p=0:plim, q=0:qlim))
bics.m <- matrix(, 1+plim, 1+qlim, dimnames=list(p=0:plim, q=0:qlim))
rmsc.m <- matrix(, 1+plim, 1+qlim, dimnames=list(p=0:plim, q=0:qlim))

# "for" loop for calculating models and stats
# Double structure needed to avoid case where "p = q = 0"

n = length(data)
```

```
44
45 for(q in 1:qlim) {
46   tmp.arima <- arima(data, c(0,2,q),optim.control = list(maxit = 1000),
47     include.mean=FALSE, method=c("ML"),seasonal=list(order=c(1,sdiff,0),period=
         season))
48   k = length(tmp.arima$coef)
49   aics.r[1, 1+q] <- tmp.arima$aic
50   aics.m[1, 1+q] <- log(tmp.arima$sigma2)+((n+2*k)/n)
51   aicc.m[1, 1+q] <- log(tmp.arima$sigma2)+((n+k)/(n-k-2))
52   bics.m[1, 1+q] <- log(tmp.arima$sigma2)+(k*log(n)/n)
53   tmp.fit <- data - tmp.arima$resid
54   tmp.fit0 <- (tmp.fit)^ip
55   rmsc.m[1, 1+q] <- sqrt(sum((data0 - tmp.fit0)^2)/(n-1))
56   }
57 for(p in 1:plim) {
58   for(q in 0:qlim) {
59     tmp.arima <- arima(data, c(p,2,q),optim.control = list(maxit = 1000),
60       include.mean=FALSE, method=c("ML"),seasonal=list(order=c(1,sdiff,0),period=
           season))
61     k = length(tmp.arima$coef)
62     aics.r[1+p, 1+q] <- tmp.arima$aic
63     aics.m[1+p, 1+q] <- log(tmp.arima$sigma2)+((n+2*k)/n)
64     aicc.m[1+p, 1+q] <- log(tmp.arima$sigma2)+((n+k)/(n-k-2))
65     bics.m[1+p, 1+q] <- log(tmp.arima$sigma2)+(k*log(n)/n)
66     tmp.fit <- data - tmp.arima$resid
67     tmp.fit0 <- (tmp.fit)^ip
68     rmsc.m[1, 1+q] <- sqrt(sum((data0 - tmp.fit0)^2)/(n-1))   }
69   }
70
71 # Look at matrices where the min value has been subtracted (makes it easier to
72 # to see which model(s) to select based on AIC, AICc, or BIC values
73 #
74 # PUT THE OUTPUT DOWN WITH OUTPUT FOR 2/3, 1/3 and FORECASTS
75 #
76 # round(aics.r - min(aics.r, na.rm=TRUE), 3)
77 # round(aics.m - min(aics.m, na.rm=TRUE), 3)
78 # round(aicc.m - min(aicc.m, na.rm=TRUE), 3)
79 # round(bics.m - min(bics.m, na.rm=TRUE), 3)
80 # round(rmsd.m - min(rmsd.m, na.rm=TRUE), 3)
81 # round(rmsc.m - min(rmsc.m, na.rm=TRUE), 3)
82
83
84 # Now repeat above, but do with 2/3 of original data and compare forecast to
85 # remaining 1/3 (using RMS values for comparison)
86
87 split <- floor((nobs-1)*2/3)+1
88 num.forecast <- (nobs) - split
89 data.train <- ts(data[1:split], start =1)
90 data.test <- data[(split+1):nobs]
91
92 data.train0 <- data0[1:split]
93 data.test0 <- data0[(split+1):nobs]
94
95
96 aics.train.r <- matrix(, 1+plim, 1+qlim, dimnames=list(p=0:plim, q=0:qlim))
97 aics.train.m <- matrix(, 1+plim, 1+qlim, dimnames=list(p=0:plim, q=0:qlim))
98 aicc.train.m <- matrix(, 1+plim, 1+qlim, dimnames=list(p=0:plim, q=0:qlim))
99 bics.train.m <- matrix(, 1+plim, 1+qlim, dimnames=list(p=0:plim, q=0:qlim))
100 rmsc.train.m <- matrix(, 1+plim, 1+qlim, dimnames=list(p=0:plim, q=0:qlim))
101 rmsc.test.m <- matrix(, 1+plim, 1+qlim, dimnames=list(p=0:plim, q=0:qlim))
102
103 # "for" loop for calculating models and stats
104 # Double structure needed to avoid case where "p = q = 0"
105
106 n.train = length(data.train)
107
108 for(q in 1:qlim) {
109   tmp.arima <- arima(data.train, c(0,2,q),optim.control = list(maxit = 1000),
```

```
    include.mean=FALSE, method=c("ML"),seasonal=list(order=c(1,sdiff,0),period=
        season))
  k = length(tmp.arima$coef)
  aics.train.r[1, 1+q] <- tmp.arima$aic
  aics.train.m[1, 1+q] <- log(tmp.arima$sigma2)+((n.train+2*k)/n.train)
  aicc.train.m[1, 1+q] <- log(tmp.arima$sigma2)+((n.train+k)/(n.train-k-2))
  bics.train.m[1, 1+q] <- log(tmp.arima$sigma2)+(k*log(n.train)/n.train)
  tmp.fit <- data.train - tmp.arima$resid
  tmp.fit0 <- (tmp.fit)^ip
  rmsc.train.m[1, 1+q] <- sqrt(sum((data.train0 - tmp.fit0)^2)/(n.train-1))
  tmp.fore <- predict(tmp.arima, n.ahead = num.forecast, se.fit = FALSE)
  tmp.fore0 <- (tmp.fore)^ip
  rmsc.test.m[1, 1+q] <- sqrt(sum((data.test0 - tmp.fore0)^2)/num.forecast)
  }
for(p in 1:plim) {
  for(q in 0:qlim) {
    tmp.arima <- arima(data.train, c(p,2,q),optim.control = list(maxit = 1000),
      include.mean=FALSE, method=c("ML"),seasonal=list(order=c(1,sdiff,0),period=
          season))
    k = length(tmp.arima$coef)
    aics.train.r[1+p, 1+q] <- tmp.arima$aic
    aics.train.m[1+p, 1+q] <- log(tmp.arima$sigma2)+((n.train+2*k)/n.train)
    aicc.train.m[1+p, 1+q] <- log(tmp.arima$sigma2)+((n.train+k)/(n.train-k-2))
    bics.train.m[1+p, 1+q] <- log(tmp.arima$sigma2)+(k*log(n.train)/n.train)
    tmp.fit <- data.train - tmp.arima$resid
    tmp.fit0 <- (tmp.fit)^ip
    rmsc.train.m[1+p, 1+q] <- sqrt(sum((data.train0 - tmp.fit0)^2)/(n.train-1))
    tmp.fore <- predict(tmp.arima, n.ahead = num.forecast, se.fit = FALSE)
    tmp.fore0 <- (tmp.fore)^ip
    rmsc.test.m[1+p, 1+q] <- sqrt(sum((data.test0 - tmp.fore0)^2)/num.forecast)
    }
  }


# Look at matrices where the min value has been subtracted (makes it easier to
# to see which model(s) to select based on AICc or BIC values

# round(aicc.m - min(aicc.m, na.rm=TRUE), 3)
# round(bics.m - min(bics.m, na.rm=TRUE), 3)


# Look at matrices where the min value has been subtracted (makes it easier to
# to see which model(s) to select based on AIC, AICc, or BIC values

round(aics.train.r - min(aics.train.r, na.rm=TRUE), 3)
round(bics.train.m - min(bics.train.m, na.rm=TRUE), 3)

# Show actual AICc, BIC, and RMS values for models and forecasts

#aicc.m
#bics.m
aics.train.r
bics.train.m
rmsc.test.m
```

Codes/ARIMA–Models–D.R

Appendix E: R code for Logistic regression models

E.1 Description

Example of R code used for Logistic regression models.

E.2 R code example for Logistic regression models

```
# R script to create logistic regression models and test them using simulation
# *** FOR MODELS BUILT FROM SINGLE YEAR'S DATA ***
# *** INDIVIDUAL MODELS -- MODEL 1
#
## LOAD NEEDED PACKAGES
library("epicalc")
library("simFrame")
## SET A RANDOM SEED VALUE SO TESTS ARE REPEATABLE?
set.seed(20081)
## READ IN DATA TO BUILD MODELS WITH
data <- read.table("data1.txt", header=TRUE, as.is=TRUE)
use(data)
## EXTRACT ROW INDEX NUMBERS FOR INCIDENTS IN DATA
inclist.fit <- which(data[,6]==1)
#### BUILD LOGISTIC REGRESSION MODELS
### (1) DURATION
system.time(glm0.single <- glm(Incident~
        Duration,
        family=binomial, data=.data))
#####
## THIS SECTION STARTS THE VALIDATION OF FORECASTS PART
#####
## READ IN DATA TO FORECAST
f.data <- read.table("data2.txt", header=TRUE, as.is=TRUE)
## CREATE LIST OF ROWS IN FORECAST DATA WITH INCIDENTS
f.inclist <- which(f.data[,6]==1)
## CREATE FORECASTS USING EACH OF THE MODELS
f0.single <- predict.glm(glm0.single, f.data, type="response")
## CREATE ARRAYS TO STORE NUMBER OF MATCHES BETWEEN FORECAST & ACTUAL
forematch0.single <- rep(NA,1000)
## DO SAMPLES AND MATCHING
system.time(for (i in 1:1000) forematch0.single[i] <- length(intersect(f.inclist,
    ups(246142, 24132,prob=f0.single))))
## SHOW SUMMARY (MEAN, STD) OF FORECAST & ACTUAL MATCHES
summ(forematch0.single)
## SAVE LIST OF FORECAST & ACTUAL MATCH COUNTS TO TEXT FILE
write(forematch0.single,file="output1.txt")
## CREATE LIST OF CUMULATIVE AVERAGES OF FORECAST & INCIDENT MATCH COUNTS
avg.forematch0.single <- cumsum(forematch0.single)/seq_along(forematch0.single)
## SAVE LIST OF CUMULATIVE AVERAGES TO TEXT FILE
write(avg.forematch0.single,file="output2.txt")
## SAVE PLOTS OF CUMULATIVE AVERAGES AS PNG FILES
png("plot.png")
plot(avg.forematch0.single)
```

```
dev.off()
### END OF COMPUTATIONS ###
q()
```

Codes/2008–09–single–1D.R

Bibliography

[1] M Braverman, J Williams, and Z Mador. Microsoft security intelligence report january-june 2006, 2006.

[2] Paul Bacher, Thorsten Holz, Markus Kotter, and Georg Wicherski. Know your enemy: Tracking botnets, 2005.

[3] US-CERT. Quarterly trends and analysis report, March" 2007.

[4] Reuters. Cybercrime is getting organized, September 2006.

[5] Anti-Phishing Working Group, April 2007.

[6] BBC News. Millions tricked by 'scareware', October 2009.

[7] BBC News. Anonymous hackers attack us security firm hbgary, February 2011.

[8] BBC News. Playstation outage caused by hacking attack, April 2011.

[9] Katherine Campbell, Lawrence A Gordon, Martin P Loeb, and Lei Zhou. The economic cost of publicly announced information security breaches: empirical evidence from the stock market. *Journal of Computer Security*, 11(3):431–448, 2003.

[10] Alessandro Acquisti, Allan Friedman, and Rahul Telang. Is there a cost to privacy breaches? an event study. In *ICIS*, page 94. Association for Information Systems, 2006.

[11] A. Shain. $25,000 password system could have halted sc hacking, November 2012.

[12] Hutton A. Hylender C. D. Pamula J. Porter C. Baker, W. and M. Spitler. Data breach: Investigations report, a study conducted by the verizon risk team with co-operation from the us secret service and the dutch high-tech crime unit, 2012.

[13] Center for Disease Control and Prevention (CDC). Key facts about seasonal flu vaccine.

[14] ISO. Information technologysecurity techniquesinformation security management systemsrequirements. ISO 27001:2005, International Organization for Standardization, Geneva, Switzerland, 2005.

[15] Gary Stoneburner, Alice Y. Goguen, and Alexis Feringa. Sp 800-30. risk management guide for information technology systems. Technical report, Gaithersburg, MD, United States, 2002.

[16] Simson Garfinkel, Gene Spafford, and Alan Schwartz. *Practical UNIX and Internet security.* " O'Reilly Media, Inc.", 2003.

[17] Robert Richardson and CSI Director. Csi computer crime and security survey. *Computer Security Institute*, 1:1–30, 2008.

[18] J. D. Howard and T. A. Longstaff. A common language for computer security incidents. *Computer*, 1998.

[19] S. Hansman and R. Hunt. A taxonomy of network and computer attacks. *Computers and Security*, 24(1):31–43, 2005.

[20] M. J. Keeling and P. Rahoni. *Modeling Infectious Diseases in Humans and Animals.* Princeton University Press, 2008. ISBN: 9781400941035.

[21] J. Kim, S. Radhakrishnan, and S. K. Dhall. Measurement and analysis of worm propagation on internet network topology. In *Computer Communications and Networks, 2004. ICCCN 2004. Proceedings 13th International Conference*, pages 495–500. IEEE, October 2004.

[22] Cliff Changchun Zou, Weibo Gong, and Don Towsley. Worm propagation modeling and analysis under dynamic quarantine defense. In *Proceedings of the 2003 ACM Workshop on Rapid Malcode*, WORM '03, pages 51–60, New York, NY, USA, 2003. ACM.

[23] AL. Goel and K. Okumoto. Time-dependent error-detection rate model for software reliability and other performance measures. *Reliability, IEEE Transactions on*, R-28(3):206–211, Aug 1979.

[24] Shigeru Yamada, Mitsuru Ohba, and S. Osaki. s-shaped software reliability growth models and their applications. *Reliability, IEEE Transactions on*, R-33(4):289–292, Oct 1984.

[25] TM Khoshgoftaar. Nonhomogeneous poisson processes for software reliability growth. In *Proc. 8th symposium in computational statistics*, pages 11–12, Copenhagen, Denmark, August 1988.

[26] J. T. Duane. Learning curve approach to reliability monitoring. *Aerospace, IEEE Transactions on*, 2(2):563–566, April 1964.

[27] Ulrich Helfenstein. Box-jenkins modelling of some viral infectious diseases. *Statistics in Medicine*, 5(1):37–47, 1986.

[28] R. Allard. Use of time-series analysis in infectious disease surveillance. *Bulletin of the World Health Organization*, 76(4):327–333, 1998.

[29] Serge Egelman, Lorrie Faith Cranor, and Jason Hong. Youve been warned: An empirical study of the effectiveness of web browser phishing warnings. In *In Proceedings of the CHI 2008 Conference on Human Factors in Computing Systems*, pages 1065–1074, Florence, Italy, 2008.

[30] P. Anbalagan and M. Vouk. Towards a unifying approach in understanding security problems. In *Software Reliability Engineering, 2009. ISSRE '09. 20th International Symposium on*, pages 136–145, Nov 2009.

[31] W. A. Arbaugh, W. L. Fithen, and J. McHugh. Windows of vulnerability: A case study analysis. *Computer*, 33(12):52–59, 2000. ISBN Number: 0018-9162.

[32] Stefan Frei, Martin May, Ulrich Fiedler, and Bernhard Plattner. Large-scale vulnerability analysis. In *Proceedings of the 2006 SIGCOMM Workshop on Large-scale Attack Defense*, LSAD '06, pages 131–138, New York, NY, USA, 2006. ACM.

[33] Julie S. Downs, Mandy B. Holbrook, and Lorrie Faith Cranor. Decision strategies and susceptibility to phishing. In *Proceedings of the Second Symposium on Usable Privacy and Security*, SOUPS '06, pages 79–90, New York, NY, USA, 2006. ACM.

[34] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer. Social phishings. *Communications of the ACM*, 50(10):94–100, 2007.

[35] Garrett Brown, Travis Howe, Micheal Ihbe, Atul Prakash, and Kevin Borders. Social networks and context-aware spam. In *In CSCW*, 2008.

[36] Ian Fette, Norman Sadeh, and Anthony Tomasic. Learning to detect phishing emails. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 649–656, New York, NY, USA, 2007. ACM.

[37] Cormac Herley. So long, and no thanks for the externalities: The rational rejection of security advice by users. In *Proceedings of the 2009 Workshop on New Security Paradigms Workshop*, NSPW '09, pages 133–144, New York, NY, USA, 2009. ACM.

[38] Nicholas Weaver, Vern Paxson, Stuart Staniford, and Robert Cunningham. A taxonomy of computer worms. In *Proceedings of the 2003 ACM Workshop on Rapid Malcode*, WORM '03, pages 11–18, New York, NY, USA, 2003. ACM.

[39] Cliff Changchun Zou, Weibo Gong, and Don Towsley. Code red worm propagation modeling and analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pages 138–147, New York, NY, USA, 2002. ACM.

[40] Z. Chen, L. Gao, and K. Kwiaty. Modeling the spread of active worms. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1890–1900. IEEE, March 2003.

[41] Stelios Sidiroglou and Angelos D. Keromytis. A network worm vaccine architecture. In *IN PROCEEDINGS OF THE IEEE WORKSHOP ON ENTERPRISE TECHNOLOGIES: INFRASTRUCTURE FOR COLLABORATIVE ENTERPRISES (WETICE), WORKSHOP ON ENTERPRISE SECURITY*, pages 220–225, 2003.

[42] Eiko Yoneki, Pan Hui, and Jon Crowcroft. Bio-inspired computing and communication. chapter Wireless Epidemic Spread in Dynamic Human Networks, pages 116–132. Springer-Verlag, Berlin, Heidelberg, 2008.

[43] J. Milliken, V. Selis, and A. Marshall. Detection and analysis of the chameleon wifi access point virus. *EURASIP J. Information Security*, 2:1–14, 2013.

[44] R. Pastor-Satorras and A Vespignani. Epidemic spreading in scale-free networks. *Phys. Rev Lett.*, 86:3200, 2001. http://dx.doi.org/10.1103/PhysRevLett.86.3200.

[45] M. Keeling. The implications of network structure for epidemic dynamics. *Theoretical Population Biology*, 67(1):1–8, February 2005.

[46] L. Danon, A. P. Ford, T. House, C. P. Jewell, M. J. Keeling, G. O. Roberts, ..., and M C. Vernon. Networks and the epidemiology of infectious disease. *Interdisciplinary Perspectives on Infectious Diseases*, *, 2011. http://dx.doi.org/10.1155/2011/284909.

[47] S. Singh, D. J. Schneider, and C. R. Myers. The structure of infectious disease outbreaks across the animal-human interface. *, *(*):*, 2013. arXiv preprint arXiv:1307.4628.

[48] A. Vazquez, B. Racz, and A. L. Barabasi. Impact of non-poissonian activity patterns on spreading processes. *Phys. Rev. Lett.*, 98(15):158702, 2007. *.

[49] S E. Schechter, J. Jung, W. Stockwell, and C. D. McLain. Inoculating ssh against address harvesting. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2006 2006*, 2006. San Diego, California, USA.

[50] C. Gao, J. Liu, and N. Zhong. Network immunization and virus propagation in email networks: Experimental evaluation and analysis. *Knowledge and Information Systems*, 27:253–279, 2011. DOI 10.1007/s10115-010-0321-0.

[51] D. N. Klaucke, J. W. Buehler, S. B. Thacker, R. G. Parrish, F. L. Trowbridge, and R. L. Berkelman. Guidelines for evaluating surveillance systems. *MMWR*, 37(s5):1–18, May 1988.

[52] P. Nsubuga, M. E. White, S. B. Thacker, M. A. Anderson, S. B. Blount, C. V. Broome, ..., and M. Trostle. *Disease Control Priorities in Developing Countries*, chapter 53. Public Health Surveillance: A Tool for Targeting and Monitoring Interventions. World Bank, Washington (DC), 2nd edition, 2006. Jamison D. T. and Breman, J. G. and Measham A. R. and et al., editors.

[53] Centers for Disease Control et al. Principles of epidemiology. *An introduction to applied epidemiology and biostatistics*, 1992.

[54] Hilary K. Browne, William A. Arbaugh, John McHugh, and William L. Fithen. A trend analysis of exploitations. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, SP '01, pages 214–, Washington, DC, USA, 2001. IEEE Computer Society.

[55] Eric Rescorla. Is finding security holes a good idea? *IEEE Security and Privacy*, 3(1):14–19, Jan 2005.

[56] O. H. Alhazmi and Y. K. Malaiya. Modeling the vulnerability discovery process. In *Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering*, ISSRE '05, pages 129–138, Washington, DC, USA, 2005. IEEE Computer Society.

[57] Omar H. Alhazmi and Yashwant K. Malaiya. Measuring and enhancing prediction capabilities of vulnerability discovery models for apache and iis http servers. In *Proceedings of the 17th International Symposium on Software Reliability Engineering*, ISSRE '06, pages 343–352, Washington, DC, USA, 2006. IEEE Computer Society.

[58] R. Anderson. Security in open versus closed systems - the dance of boltzmann, coase and moore. In *Open source software: Economics, Law and Policy*, June 20-21 2002. Toulouse, France.

[59] D. J. Watts, R. Muhamad, D. C. Medina, and P. S. Dodds. Multiscale, resurgent epidemics in a hierarchical metapopulation model. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 102, pages 11157–11162, 2005. doi: 10.1073/pnas.0501226102.

[60] H. Trottier, P. Philippe, and R. Roy. Stochastic modeling of empirical time series of childhood infectious diseases data before and after mass vaccination. *Emerging Themes in Epidemiology*, 3(1):9, 2006.

[61] Dejian Lai. Monitoring the sars epidemic in china: a time series analysis. *Journal of Data Science*, 3(3):279–293, 2005.

[62] B. Han and T. Y. Leong. We did the right thing: An intervention analysis approach to modeling intervened sars propagation in singapore. *Studies in Health Technology and Informatics*, 107(Part 2):1246–1250, 2004.

[63] E. Michael, M. N. Malecela-Lazaro, P. E. Simonsen, E. M. Pedersen, G. Barker, A. Kumar, and J. W. Kazura. Mathematical modelling and the control of lymphatic filariasis. *The Lancet Infectious Diseases*, 4(4):223–234, 2004.

[64] J. W. Hay and J. I. Ward. Economic considerations for pertussis booster vaccination in adolescents. *Pediatr. Infect. Dis. J.*, 24(6):S127–S133, June 2005.

[65] C. Hoggart, P. Brennan, A. Tjonneland, U. Vogel, K. Overvad, J. N. Ostergaard, ..., and P. Vineis. A risk model for lung cancer incidence. *Cancer Prev. Res. (Phila)*, 5(6):834–846, June 2012.

[66] James C Frauenthal. *Mathematical modeling in epidemiology*. Universitext. Springer, Berlin, 1980.

[67] Bruce Rogers. Epidemic models on networks: Space, the final frontier, 2010.

[68] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014.

[69] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[70] Cliff C Zou, Don Towsley, and Weibo Gong. Email virus propagation modeling and analysis. *Department of Electrical and Computer Engineering, Univ. Massachusetts, Amherst, Technical Report: TR-CSE-03-04*, 2003.

[71] Alexander Mintz and Milton L Blum. A re-examination of the accident proneness concept. *Journal of Applied Psychology*, 33(3):195–211, 1949.

[72] J.C. Laprie and K. Kanoun. Trend analysis. In Michael R. Lyu, editor, *Handbook of Software Reliability Engineering*, pages 401–438. IEEE Computer Society Press, Los Alamitos, CA, USA, 1996.

[73] Karama Kanoun, Marta Rettelbusch de Bastos Martini, and Jorge Moreira de Souza. A method for software reliability analysis and prediction application to the tropico-r switching system. *Software Engineering, IEEE Transactions on*, 17(4):334–344, 1991.

[74] Karama Kanoun, Mohamed Kaâniche, and J-C Laprie. Qualitative and quantitative reliability assessment. *Software, IEEE*, 14(2):77–87, 1997.

[75] G.J. Knafl. Solving maximum likelihood equations for two-parameter software reliability models using grouped data. In *Software Reliability Engineering, 1992. Proceedings., Third International Symposium on*, pages 205–213, Oct 1992.

[76] Robert S Pindyck and Daniel L Rubinfeld. *Econometric models and economic forecasts*. McGraw-Hill Book Company, New York, 1981.

[77] G. Box and G. Jenkins. *Time series analysis: forecasting and control*. Holden Day, San Francisco, CA, 1970.

[78] Peter J Brockwell and Richard A Davis. *Time series: theory and methods*. Springer-Verlag, New York, 1991.

[79] Robert H Shumway and David S Stoffer. *Time series analysis and its applications*. Springer-Verlag, New York, 2000.

[80] H Akaike. Information theory and an extension of the maximum likelihood principle. In BN Petran and F Csáki, editors, *International symposium on information theory, Second edition*, pages 267–281, Akadeemiai Kiadi, Budapest, Hungary, 1973.

[81] Clifford M Hurvich and Chih-Ling Tsai. Regression and time series model selection in small samples. *Biometrika*, 76(2):297–307, 1989.

[82] Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[83] Martin Roesch et al. Snort: Lightweight intrusion detection for networks. In *LISA*, volume 99, pages 229–238, 1999.

[84] Mark A Beaumont, Wenyang Zhang, and David J Balding. Approximate bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002.

[85] Michael GB Blum and Viet Chi Tran. Hiv with contact tracing: a case study in approximate bayesian computation. *Biostatistics*, 11(4):644–660, 2010.

[86] Tina Toni, David Welch, Natalja Strelkowa, Andreas Ipsen, and Michael P.H Stumpf. Approximate bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of The Royal Society Interface*, 6(31):187–202, 2009.

[87] Richard R Picard and R Dennis Cook. Cross-validation of regression models. *Journal of the American Statistical Association*, 79(387):575–583, 1984.

[88] Edward J Hannan and Barry G Quinn. The determination of the order of an autoregression. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 190–195, 1979.

[89] Edward Condon, Michel Cukier, and Tao He. Applying software reliability models on security incidents. In *Proceedings of the The 18th IEEE International Symposium on Software Reliability*, pages 159–168. IEEE Computer Society, 2007.

[90] M Xie and SL Ho. Analysis of repairable system failure data using time series models. *Journal of Quality in Maintenance Engineering*, 5(1):50–61, 1999.

[91] Tyler Moore, Richard Clayton, and Ross Anderson. The economics of online crime. *The Journal of Economic Perspectives*, 23(3):3–20, 2009-08-01T00:00:00.

[92] Jason Franklin, Adrian Perrig, Vern Paxson, and Stefan Savage. An inquiry into the nature and causes of the wealth of internet miscreants. In *ACM conference on Computer and communications security*, pages 375–388, 2007.

[93] Danielle Chrun. *Model-Based Support for Information Technology Security Decision Making*. PhD thesis, University of Maryland, College Park, 2011.

[94] Virasakdi Chongsuvivatwong. *Analysis of epidemiological data using R and Epicalc*. Chanmuang Press, Songkhla, Thailand, 2008.

[95] Margaret Sullivan Pepe and Todd A Alonzo. Comparing disease screening tests when true disease status is ascertained only for screen positives. *Biostatistics*, 2(3):249–260, 2001.

[96] James A Hanley and Barbara J McNeil. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology*, 148(3):839–843, 1983.