Modeling and Automated Containment of Worms

Sarah H. Sellke, *Student Member*, *IEEE*, Ness B. Shroff, *Fellow*, *IEEE*, and Saurabh Bagchi, *Senior Member*, *IEEE*

Abstract—Self-propagating codes, called *worms*, such as Code Red, Nimda, and Slammer, have drawn significant attention due to their enormously adverse impact on the Internet. Thus, there is great interest in the research community in modeling the spread of worms and in providing adequate defense mechanisms against them. In this paper, we present a (stochastic) *branching process* model for characterizing the propagation of Internet worms. The model is developed for uniform scanning worms and then extended to preference scanning worms. This model leads to the development of an automatic worm containment strategy that prevents the spread of a worm beyond its early stage. Specifically, for uniform scanning worms, we are able to 1) provide a precise condition that determines whether the worm spread will eventually stop and 2) obtain the distribution of the total number of hosts that the worm infects. We then extend our results to contain preference scanning worms. Our strategy is based on limiting the number of scans to dark-address space. The limiting value is determined by our analysis. Our automatic worm containment schemes effectively contain both uniform scanning worms and local preference scanning worms, and it is validated through simulations and real trace data to be nonintrusive. We also show that our worm strategy, when used with traditional firewalls, can be deployed incrementally to provide worm containment for the local network and benefit the Internet.

Index Terms—Internet scanning worms, stochastic worm modeling, branching process model, preference scanning worms, automatic worm containment.

1 Introduction

The Internet has become critically important to the financial viability of the national and the global economy. Meanwhile, we are witnessing an upsurge in the incidents of malicious code in the form of computer viruses and worms. One class of such malicious code, known as random scanning worms, spreads itself without human intervention by using a scanning strategy to find vulnerable hosts to infect. Code Red, SQL Slammer, and Sasser are some of the more famous examples of worms that have caused considerable damage. Network worms have the potential to infect many vulnerable hosts on the Internet before human countermeasures take place. The aggressive scanning traffic generated by the infected hosts has caused network congestion, equipment failure, and blocking of physical facilities such as subway stations, 911 call centers, etc. As a representative example, consider the Code Red worm Version 2 that exploited a buffer overflow vulnerability in the Microsoft IIS Web servers. It was released on 19 July 2001 and over a period of less than 14 hours infected more than 359,000 machines. The cost of the epidemic, including subsequent strains of Code Red, has been estimated by Computer Economics to be $2.6 billion [34]. Although Code Red was particularly virulent in its economic impact (for example, see [2] and [22]), it provides an indication of the magnitude of the damage that can be inflicted by such worms. Thus, there is a need to carefully characterize the spread of worms and develop efficient strategies for worm containment.

The goal of our research is to provide a model for the propagation of random scanning worms and the corresponding development of automatic containment mechanisms that prevent the spread of worms beyond their early stages. This containment scheme is then extended to protect an enterprise network from a preference scanning worm. A host infected with random scanning worms finds and infects other vulnerable hosts by scanning a list of randomly generated IP addresses. Worms using other strategies to find vulnerable hosts to infect are not within the scope of this work. Some examples of nonrandom-scanning worms are e-mail worms, peer-to-peer worms, and worms that search the local host for addresses to scan.

Most models of Internet-scale worm propagation are based on deterministic epidemic models [7], [28], [31]. They are acceptable for modeling worm propagation when the number of infected hosts is large. However, it is generally accepted that they are inadequate to model the early phase of worm propagation accurately because the number of infected hosts early on is very small [20]. The reason is that epidemic models capture only expected or mean behavior while not being able to capture the variability around this mean, which could be especially dramatic during the early phase of worm propagation. Although stochastic epidemic models can be used to model this early phase, they are generally too complex to provide useful analytical solutions.

- S.H. Sellke and S. Bagchi are with the School of Electrical and Computer Engineering, Purdue University, Engineering Building, 465 Northwestern Ave., West Lafayette, IN 47907. E-mail: {ssellke, sbagchi}@purdue.edu.
- N.B. Shroff is with the Department of Electrical Engineering, The Ohio State University, 2015 Neil Ave., Columbus, OH 43210. E-mail: shroff@ece.osu.edu.

Manuscript received 17 Sept. 2006; revised 11 Mar. 2007; accepted 21 Aug. 2007; published online 13 Sept. 2007.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-0133-0906. Digital Object Identifier no. 10.1109/TDSC.2007.70230.

In this paper, we propose a stochastic branching process model for the early phase of worm propagation.[1] We consider the generation-wise evolution of worms, with the hosts that are infected at the beginning of the propagation forming generation zero. The hosts that are directly infected by hosts in generation n are said to belong to generation $n + 1$. Our model captures the worm spreading dynamics for worms of arbitrary scanning rate, including stealth worms that may turn themselves off at times.

We show that it is the total number of scans that an infected host attempts, and not the more restrictive scanning rate, which determines whether worms can spread. Moreover, we can probabilistically bound the total number of infected hosts. These insights lead us to develop an automatic worm containment strategy. The main idea is to limit the total number of distinct IP addresses contacted (denote the limit as M_C) per host over a period we call the *containment cycle*, which is of the order of weeks or months. We show that the value of M_C does not need to be as carefully tuned as in the traditional rate control mechanisms. Further, we show that this scheme will have only marginal impact on the normal operation of the networks. Our scheme is fundamentally different from rate limiting schemes because we are not bounding instantaneous scanning rates.

Preference scanning worms are a common class of worms but have received significantly less attention from the research community. Unlike uniform scanning worms, this type of worm prefers to scan random IP addresses in the local network to the overall Internet. We show that a direct application of the containment strategy for uniform scanning worms to the case of preference scanning worms makes the system too restrictive in terms of the number of allowable scans from a host. We therefore propose a local worm containment system based on restricting a host's total number of scans to local unused IP addresses (denoted as N). We then use a stochastic branching process model to come up with a bound on the value of N to ensure that the worm spread is stopped.

The main contributions of the paper are summarized as follows: We provide a means to accurately model the early phase of propagation of uniform scanning worms. We also provide an equation that lets a system designer probabilistically bound the total number of infected hosts in a worm epidemic. The parameter that controls the spread is the number of allowable scans for any host. The insight from our model provides us with a mechanism for containing both fast-scanning worms and slow-scanning worms without knowing the worm signature in advance or needing to detect whether a host is infected. This scheme is nonintrusive in terms of its impact on legitimate traffic. Our model and containment scheme is validated through analysis, simulation, and real traffic statistics.

The rest of the paper is organized as follows: In Section 2, we review relevant research on network worms. In Section 3, we present our *branching process* model with corresponding analytical results on the spread of the infection. In Sections 4 and 5, we describe an automatic worm containment scheme for random scanning worms and adaptation to the case of local preference scanning worms. In Section 6, we provide numerical results that validate our model and confirm the effectiveness of our containment scheme. In Section 7, we summarize our contributions and provide some discussion and directions for future work.

1. Branching process approximations to stochastic epidemic processes have been studied rigorously. A tutorial on this topic can be found in [4].

2 Related Work

As mentioned in Section 1, deterministic epidemic models have been used to study worm propagation [28], [31]. For illustration, consider the two-factor worm model proposed by Zou et al. [31]:

$$\frac{dI(t)}{dt} = \beta(t)[V - R(t) - I(t) - Q(t)]I(t) - \frac{dR(t)}{dt}, \qquad (1)$$

where V is the total number of susceptible hosts on the Internet, and $I(t)$, $R(t)$, $Q(t)$ represent the number of infectious hosts, the number of removed hosts from the infectious population, and the number of removed hosts from the susceptible population at time t, respectively. The parameter $\beta(t)$ is the infection rate at time t and reflects the impact of the Internet traffic on the worm propagation. The parameters $R(t)$ and $Q(t)$ reflect the human countermeasures in patching.

When there is no patching and when the infection rate is constant, the two factor model equation is the random constant spread (RCS) model proposed by Staniford et al. [28]:

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

These types of models are suitable when there are a large number of infected hosts. However, during the early stage of the worm propagation, the number of infected hosts is small and such deterministic models may not accurately characterize the spread of worms. Nonetheless, most existing models for Internet worms are based on deterministic epidemic models.

Early worm detection systems have been proposed by several researchers. Zou et al. use a Kalman filter [32] to detect the worms. The Kalman filter is used to detect the presence of a worm by detecting the trend, not the rate, of the observed illegitimate scan traffic. The filter is used to separate worm traffic from background nonworm scan traffic. Liljenstam et al. [20] and Berk et al. [6] develop an early worm detection system called DIB:S/TRAFEN in which a select group of routers forward ICMP T-3 packets to the analysis station. It is shown in [20] that the total number of participating routers can be small, but these routers must be distributed across a significant fraction of the Internet address space to ensure timely and accurate worm detection. They develop a worm simulation model that is used for generating worm traffic for evaluating the DIB:S/TRAFEN detection system. Their simulation model uses a combination of the deterministic epidemic model and a general stochastic epidemic model to model the effect of large-scale worm attacks. They found the stochastic epidemic model to be useful for modeling the early stage of the worm spread. However, the complexity of the general stochastic epidemic model makes it difficult to derive insightful results that could be used to contain the worm.

Rate-control-based countermeasures, such as *Virus throttling* by Williamson [29], have been shown to be successful in detecting and slowing down fast scanning worms. Wong et al. [30] studied the effect of rate control on suppressing the spread of the worms when this mechanism is deployed at various points (for example, host, LAN, and core router) of the network. The rate control is effective in slowing down fast worms but is not effective against slow scanning worms. In addition, the limit on the rate must be carefully tuned in order to let the normal traffic through.

Zou et al. propose and analyze a dynamic quarantine scheme for Internet worms [33]. They assume that the underlying worm detection system has a certain false alarm rate. Their system confines all hosts that have triggered the alarm and automatically releases them after a short time. They found that this scheme can slow down the worm spread but cannot guarantee containment.

Moore et al. [24] examined the reaction time required to contain Internet-scale worms using countermeasures such as blacklisting the infected hosts and content filtering at routers. Their study concluded that to effectively contain Internet worms, it is necessary to take actions early, within minutes of the worm outbreak.

Ganesh et al. investigated how topology affects the spread of an epidemic by modeling the epidemic spread as a contact process in a finite undirected graph [10]. Most recently, Ganesh et al. proposed an interesting framework [11] to examine the impact of the worm countermeasures such as worm throttling and worm quarantine on the spread of the worms. The interaction between the worm detection strategy and worm propagation is viewed as a game. Based on their analysis, they designed optimal detection rules against scanning worms and further proposed methods of coordinating information among the end hosts to speed up the detection using the Bayesian decision theory.

Another approach to worm detection is based on *Sequential Hypothesis Testing* [14], [41], [45]. To determine if a host is infected, a sequence of connection attempts, both successful and failed, is examined to validate the null hypothesis that the host is not infected. If a host is deemed infected, the scans from it are suppressed. The shortcoming is that the infected hosts may avoid detection if they fake more successful connections.

The effect of NAT in the spread of local preference scanning worms is evaluated in [38]. The authors found that the increased deployment of NAT can slow down the spread of the LPS worms. They pointed out that the decreased vulnerability density inside NAT is responsible for the slowing spread. In [37], Rajab et al. investigated a distributed monitor for worm detection. They found the vulnerable hosts follow a power law distribution, and LPS worms can spread much faster than the uniform scanning worms in this situation. They also show that the judicious placement of distributed monitors for worms is more effective than centralized monitors in terms of the speed of worm containment for local preference scanning worms.

3 Branching Process Model for Random Scanning Worms

We now present the branching process model we use to characterize the propagation of random scanning worms. Scanning worms are those that generate a list of random IP addresses to scan from an infected host. The uniform scanning worms are those in which the addresses are chosen completely randomly while preference scanning worms weigh the probability of choosing an address from different parts of the network differently. In this paper, we first describe the approach for uniform scanning worms, and then, we present the extension for preference scanning worms. In our model, a host under consideration is assumed to be in one of three states: *susceptible*, *infected*, or *removed*. An infected host generates a list of random IP address to scan. If a susceptible host is found among the scans, it will become infected. A removed host is one that has been removed from the list of hosts that can be infected.

We use V to denote the total number of initially vulnerable hosts. The initial probability of successfully finding a vulnerable host in one scan is $p = \frac{V}{2^{32}}$, where 2^{32} is the size of current IPv4 address space. We call p the density of the vulnerable hosts or *vulnerability density*.

We use M to denote the total number of scans from an infected host. This M is the "natural" limit of the worm itself and is always finite during a finite period of time. For example, when a worm scans six IPs per second, it will scan $M = 518,400$ times in a day. We will characterize the values of M that ensure extinction of a worm in the Internet and provide the probability distribution of the total number of infected hosts as a function of M. To that end, we first describe our branching process model.

3.1 Galton-Watson Branching Process

The Galton-Watson Branching process[2] is a Markov process that models a population in which each individual in generation n independently produces some random number of individuals in generation $n+1$, according to a fixed probability distribution that does not vary from individual to individual [15], [27].

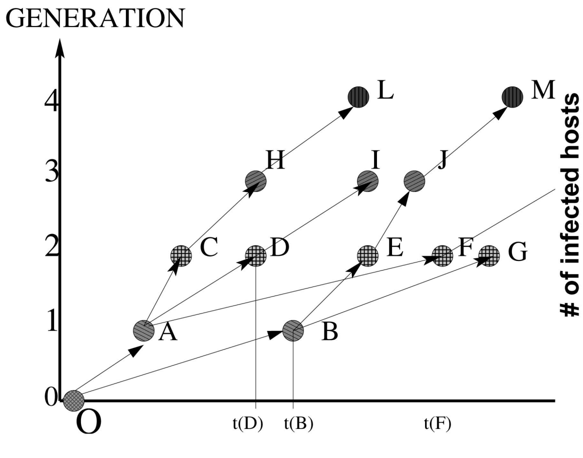
All infected hosts can be classified into generations in the following manner. The initially infected hosts belong to the 0th generation. All hosts that are directly infected by the initially infected hosts are 1st generation hosts, regardless of when they are infected. In general, an infected host H_b is an $(n+1)\text{st}$ generation host if it is infected *directly* by a host H_a from the $n\text{th}$ generation. H_b is also called an offspring of H_a. All infected hosts form a tree if we draw a link between a host and its offspring. Fig. 1a illustrates the notion of generationwise evolution. In this model, there is no direct relationship between generation and time. A host in a higher generation may precede a host in a lower generation, as host D (generation 2) precedes host B (generation 1) in Fig. 1a $(t(D) < t(B))$. Fig. 1b illustrates the Code Red propagation in the early stage showing the growth of the number of infected hosts of the first six generations.

Let ξ be the random variable representing the offsprings of (that is, the number of vulnerable hosts infected by) one infected host scanning M times. During the early phase of the propagation, the vulnerability density p remains constant since the number of infected hosts is much smaller than the number of vulnerable hosts in the population. Thus, during the initial phase of the worm propagation, ξ is a $binomial(M, p)$ random variable. Hence,

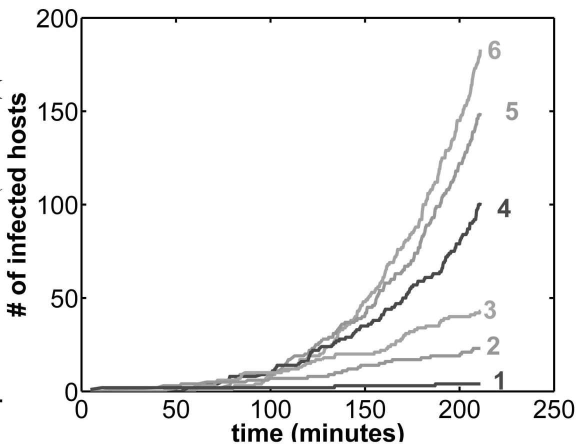
$$P\{\xi = k\} = \binom{M}{k} p^k (1-p)^{M-k}, \qquad k = 0, 1, \cdots, M. \qquad (2)$$

Let I_n be the number of infected hosts in the $n\text{th}$ generation. I_0 is the number of initial hosts that are infected. During early phase of worm propagation, each infected host in the $n\text{th}$ generation infects a random number of vulnerable hosts, independent of one another, according to the same probability distribution. These newly infected hosts are the

2. Branching process models have already been successfully used in modeling the spread of infectious diseases in the early phase of the outbreak [4].



(a)



(b)

Fig. 1. (a) Generationwise evolution in a tree structure, with O the initially infected host. O has two offsprings: host A and host B. (b) Growth of the infected hosts in generations. The generation number is shown next to the growth curve.

$(n+1)$st generation hosts. Let $\xi_k^{(n)}$ denote the number of hosts infected by the kth infected host in the nth generation. The number of infected hosts in the $(n+1)$st generation can be expressed as $I_{n+1} = \sum_{k=1}^{I_n} \xi_k^{(n)}$, where $\xi_k^{(n)}$ are independent $binomial(M, p)$ random variables.

During the initial worm epidemic, each infected host produces offsprings independently and according to the same probability distribution, as in (2). Therefore, the spread of infected hosts in each generation $\{I_n, n \geq 0\}$ forms a branching process. The branching process accurately models the early phase of worm propagation. When the worm propagation is beyond the early phase, the branching process model gives an upper bound on the spread of worms. This is because the vulnerability density is smaller, and the probability of two infected hosts scanning the same vulnerable host is higher later on. Thus, if we can ensure that the branching process does not spread, it also ensures that the worm propagation is contained.

For convenience, we provide a list of the symbols used in our model and their corresponding definitions in Table 1.

We next use the branching process model to answer questions on how the worm propagates as a function of the total number of allowable scans M.

3.2 Extinction Probability for Scanning Worms

We model worm propagation as a branching process $\{I_n\}_{n=0}^{\infty}$, where I_n is the number of infected hosts in the nth generation. The *extinction probability* of a branching process $\{I_n\}_{n=0}^{\infty}$ is the probability that the population dies out eventually. It is defined as

$$\pi = P\{I_n = 0, \quad \text{for some } n\}. \tag{3}$$

When the branching process model is used for the spread of random scanning worms, the extinction probability measures the likelihood of the worm dying out after a certain number of generations. When $\pi = 1$, we are certain that the infections from the worm cannot be spread for an arbitrarily large number of generations. The following *Proposition* provides the necessary and sufficient condition for worm extinction.

Proposition 1. *Let $\{I_n\}_{n=0}^{n=\infty}$ be a branching process model for worm propagation, where I_n models the number of infected hosts in the nth generation. Let p be the density of vulnerable hosts and M be the total number of scans attempted by an infected host. Then, $\pi = 1$ if and only if $M \leq \frac{1}{p}$.*

Proof. According to Theorem 4.5.1 in [27], the extinction probability of a branching process is 1 if and only if the expected number of offsprings from each individual is no more than 1.

Let ξ be the random variable representing the number of offsprings produced by an infected host. ξ is a Binomial random variable with parameters (M, p). The expected value of ξ is $E(\xi) = Mp$. By Theorem 4.5.1 in [27], $\pi = 1$ if and only if $E(\xi) \leq 1$. Therefore, $\pi = 1$ if and only if $M \leq \frac{1}{p}$. □

Since $E[\xi] = Mp$, Mp is the *basic reproduction number*, R_0, in the epidemiology literature.

Using *Code Red* and *SQL Slammer* as examples, if M is no more than 11,930 and 35,791, respectively, the worms would eventually die out.[3] The value of M corresponds to the number of unique addresses that can be contacted and, therefore, the restriction on M is not expected to significantly interfere with normal user activities. This is borne out by actual data for traffic originated by hosts at the Lawrence Berkeley National Laboratory and Bell Labs presented in Section 4.

If M is different for different hosts, we can use multitype branching processes to obtain conditions under which the extinction probability is 1. If r and $1-r$ are the fractions of hosts that scan M_1 and M_2 times, respectively, the extinction probability is 1 if $M_1 r + M_2(1-r) < 1/p$.

3.3 Probability Distribution of Total Infections

Although the probability of extinction gives us a bound on the maximum number of allowable scans per host, the true effectiveness of a worm containment strategy is measured by *how many hosts are infected before the worm finally dies out*. We next provide a probability density function for the total number of infections when the total number of scans per host is below $1/p$. The probability density function is applicable only when the total number of scans per host is below $1/p$. It has as a parameter M, which is typically kept below $1/p$.

The total number of infections, denoted by I, is the sum of the infections in all generations ($I = \sum_{n=0}^{\infty} I_n$). Our

3. $V = 360,000$ for Code Red and $V = 120,000$ for SQL Slammer are used in this calculation.

TABLE 1
Notations

Symbol	Explanation
V	The size of the uninfected host population.
p	The vulnerability density, specified in terms of entire network address space, used and unused. e.g. $p = \frac{V}{2^{32}}$ for IPv4.
M	The number of scans from a host
M_C	The maximum number of scans allowed from a host within a containment cycle.
ξ	Random number of offsprings generated by each infected host
$\xi_k^{(n)}$	Number of offsprings produced by k^{th} host in n^{th} generation
I_0	Number of initially infected hosts
I_n	Number of n^{th} generation infected hosts
I	Total number of all infected hosts $[I = \sum_{n=0}^{\infty} I_n]$
π	Extinction probability
P_n	Extinction probability at n^{th} generation, i.e.,$P[I_n = 0]$

objective is to provide a simple closed-form equation that accurately characterizes $P\{I = k\}$, the probability that the total number of hosts infected is k, for a given value of M.

We consider any uniform scanning worm with I_0 initially infected hosts. We allow all hosts to scan $M \leq 1/p$ times, where the vulnerability density is p, and the total number of infected hosts is $I = \sum_{n=0}^{\infty} I_n$. As stated earlier, $\{I_n\}$ is a branching process. The infected hosts independently infect a random number of vulnerable hosts that obeys the same probability distribution as ξ. Since the total number of scans per infected host is M, ξ is a binomial random variable $B(M, p)$. Further, since p is typically small in practice (for example, $p \approx 8.4 \times 10^{-4}$ for Code Red), and M is typically large, the probability distribution of ξ can be accurately approximated by a Poisson distribution with mean $\lambda = Mp$. Hence, the probability density function for ξ is

$$P\{\xi = k\} \approx e^{-\lambda} \frac{(\lambda)^k}{k!}.$$

It then follows directly in [8] that the total progeny of the branching process has a Borel-Tanner distribution, that is,

$$P\{I = k\} = \frac{I_0}{k(k - I_0)!} (k\lambda)^{(k-I_0)} e^{-k\lambda}, \quad k = I_0, I_0 + 1, \cdots, \tag{4}$$

where $\lambda = Mp$. The mean and variance of I are given in [8]

$$E(I) = \frac{I_0}{1 - \lambda} \quad VAR(I) = \frac{I_0}{(1 - \lambda)^3}.$$

More importantly,

$$p_k = P\{I \leq k\} = \sum_{i=I_0}^{k} P\{I = i\} = \sum_{i=I_0}^{k} \frac{I_0}{i(i - I_0)!} (i\lambda)^{(i-I_0)} e^{-i\lambda}, \quad k = I_0, I_0 + 1, \cdots. \tag{5}$$

Equation (5) for $p_k = P\{I \leq k\}$ is important because it probabilistically determines the spread of the worm for a given limit on the number of scans M. Thus, p_k can be used to design a containment strategy that can be tuned to specifications.

Consider Code Red with 10 initially infected hosts, as used in [33]. If we would like $p_{360} = P\{I \leq 360\} = 0.99$, we could use (5) to set $M = 10,000$ ($\lambda = Mp = 0.83$). That is, if we want the total number of infected hosts to be less than 360 with probability 0.99, we set $M = 10,000$ to achieve this.

Fig. 2a shows the plot of the probability density function of I for three different values of M for Code Red with 10 initial infections. Fig. 2b plots the cumulative probability distribution of I for three different values of M for Code Red with 10 initial infections. As we can see in Fig. 2b, with high probability (0.95), Code Red will not spread to more than 150, 50, and 27 total infected hosts if the values of M are chosen to be 10,000, 7,500, and 5,000, respectively.

We also consider the SQL Slammer worm with 10 initial infected hosts. If we use the same value for M ($M = 10,000$), with high probability (0.97), no more than 20 vulnerable hosts (or 10 additional vulnerable hosts) will be infected. This corresponds to 0.008 percent of the total vulnerable population. If we further reduce M to 5,000, with high probability (0.97), no more than four additional vulnerable hosts will be infected.

Now, we compare this result to existing worm detection systems [20], which provide detection when approximately 0.03 percent (Code Red) and 0.005 percent (slammer) of the susceptible hosts are infected. This performance is achieved by careful selection of the routers at which worm detection mechanisms are put in place. With our scheme, when M is kept below a predefined threshold, with very high probability, the infection will not be allowed to spread that widely. Further, our results also hold for slow worms, which most other detection techniques, including [20], have trouble detecting.

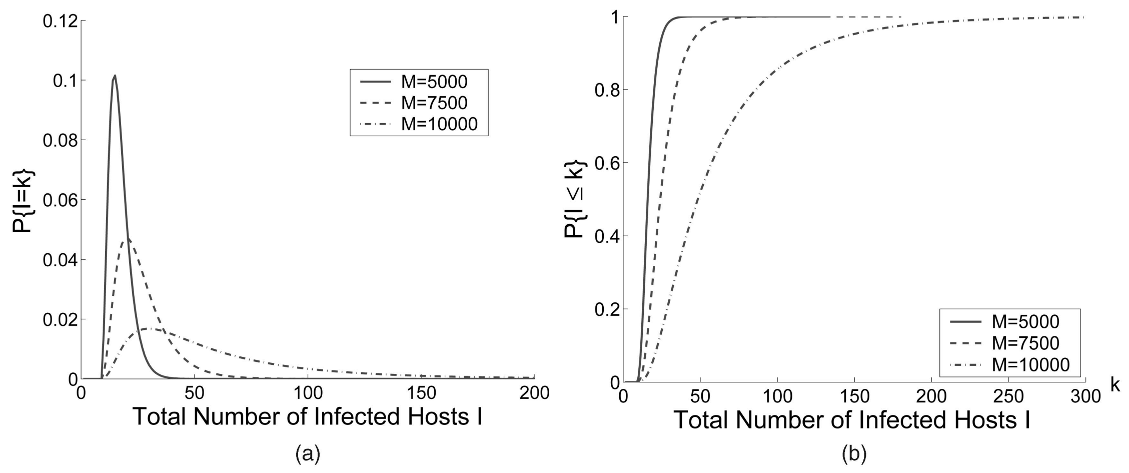


Fig. 2. Probability density and cumulative distribution of I, the total number of infected hosts.

Based on our analysis, we now develop an automatic worm containment system that can inhibit the spread of the worms.

4 Automated Worm Containment System

The results in Section 3 provide us with a blueprint of a *host-based* worm containment strategy. The containment system is based on the idea of restricting the total number of scans to unique IP addresses by any host. We assume that we can estimate or bound the percentage of infected hosts in our system. Our proposed automated worm containment strategy has the following steps:

1. Let M_C be the total number of unique IP addresses that a host can contact in a *containment cycle*. At the beginning of each new containment cycle, set a counter that counts the number of unique IP addresses for each host to be zero.
2. Increment this counter for each host when it scans a new IP address.
3. If a host reaches its scan limit before the end of the containment cycle, it is removed and goes through a heavy-duty checking process to ensure that it is free of infection before being allowed back into the system. When allowed back into the system, its counter is reset to zero.
4. Hosts are thoroughly checked for infection at the end of a containment cycle (one by one to limit the disruption to the network) and their counters reset to zero.

Choose M_C to probabilistically bound the total number of infected hosts (I) to less than some acceptable value ϵ, (for example, $p_\epsilon = P\{I \leq \epsilon\} \geq 0.99$), as given by (5). Further, the containment cycle can be obtained through a learning process. For example, initially, one could choose a containment cycle of a fixed but relatively long duration, for example, a month. Since the value of M_C that we can allow is fairly large (on the order of thousands, as indicated by analysis with SQL Slammer and Code Red), we do not expect that normal hosts will be impacted by such a restriction. We can then increase (or, for the rare hosts, decrease) the duration of the containment cycle depending on the observed activity of scans generated by correctly operating hosts. Also, the containment cycle is determined to ensure that the number of scans from legitimate hosts is highly probable to be less than M_C. Since typical values of M_C are large (that is, most legitimate hosts will not reach this value even over a month's time frame), the containment cycle is quite large. Typically, computer security patches are pushed out weekly or biweekly and machines are brought down during the patching process. For example, Purdue's engineering machines are brought down once a week. The worm containment cycle check can be done during this maintenance period.

The "heavy duty checking" in *Step 3* could even include human intervention. Since the number of offending hosts is small, administrators should be able to take the machine offline and perform a thorough checking. The first step of the heavy-duty checking should be to follow a common security best-practice procedure. For example, one must make sure that the antivirus software is up-to-date and is not disabled. One also needs to run a file integrity checker to make sure that the critical files are not modified, and no new executables are installed. After routine checking with all the available tools, an experienced system administrator should be able to make a final decision as to whether or not to let this machine be back online. Fig. 3 illustrates our worm containment scheme.

We use the 30 day trace of wide-area TCP connections (LBL-CONN-7) [35] originating from 1,645 hosts in the Lawrence Berkeley Laboratory to analyze the growth of the number of unique destination IP addresses per host (this is clean data over a period when there was no known worm traffic in the network). Our study indicates that 97 percent of hosts contacted less than 100 distinct destination IP addresses during this period. Only six hosts contacted more than 1,000 distinct IP addresses, and the most active host contacted approximately 4,000 unique IP addresses. Fig. 4a shows the growth trend of the total unique destination IP addresses for these six most active hosts.

We also analyze HTTP traffic using more recent trace data from NLANR [25]. The trace data consists of contiguous Internet access IP headers collected at Bell Labs during one week in May 2002. The network serves about 400 people. We find that there are three Web proxies that each has contacted a little over 20K Web servers. The Web servers being contacted are highly correlated. The total number of distinct Web servers contacted by all the three proxies is 46K, not 60K. This

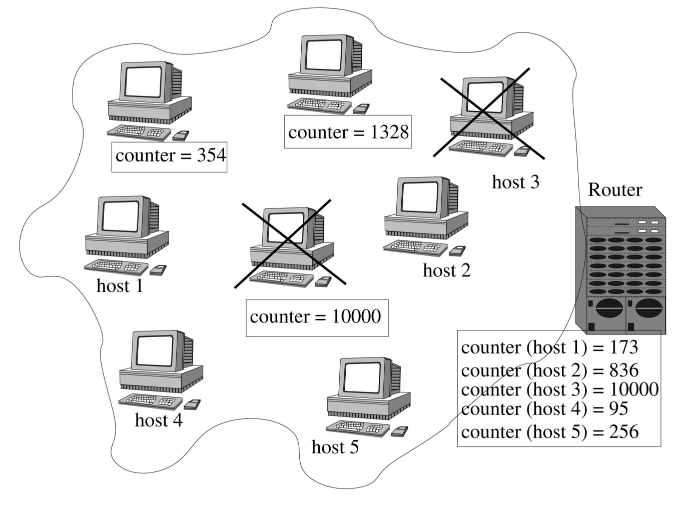


Fig. 3. Worm containment system for the uniform scanning worms. M_C is set to 10,000. The total number of scans for each host is monitored. The monitoring system can be implemented on each host or on the edge router of a local network. The two hosts marked are removed from the network automatically because their total number of scans (counter) has reached 10,000.

correlation may aid in reducing the storage requirement and correspondingly speeding the search performance when the unique destinations are maintained.

The data on the traffic between the internal hosts and the Web proxies is not available. There are two other hosts, which contacted between 5K and 6K Web servers. The rest of the hosts contacted only a few hundred Web servers. The histogram of those hosts (excluding the three proxies and two others mentioned) is shown in Fig. 4b.

If our containment system is used with the containment cycle to be one week and M_C is set to be 5,000, none of the above hosts (except for the five mentioned above) will trigger an alarm. As shown in Section 3, with high probability, the total infections caused by Code Red will be under 27 hosts when $M = 5,000$. *This suggests that our containment system is not likely to interfere significantly with normal traffic, yet it contains the spread of the worms*. If our scheme is implemented in a network with proxies, the best solution would be to have the proxies do the counting of scans on an individual host basis.

The containment cycle can also be adaptive and dependent on the scanning rate of a host. If the number of scans originating from a host gets close to the threshold, say, it reaches a certain fraction f of the threshold, then the host goes through a complete checking process. The advantage of this worm containment system is that it does not depend on diagnosis of infected hosts over small time granularities. It is also effective in preventing an Internet scale worm outbreak because the total number of infected hosts is extremely low, as shown in the examples in the previous section.

Traditional rate-based techniques attempt to limit the spread of worms by limiting the scanning rate. The limit imposed must be carefully tuned so as to not interfere with normal traffic. For example, the *rate throttling* technique [29] limits the host scan rate to 1 scan per second. The rate limiter can inhibit the spread of fast worms without interfering with normal user activities. However, slow scanning worms with scanning rate below 1 Hz and stealth worms that may turn themselves off at times will elude detection and spread slowly.

In contrast, our worm containment system can contain fast worms, slow worms, and stealth worms. The fast scanning worms will reach the limit on M_C sooner, whereas the slow worms will reach this limit after a longer period of time. As long as the host is disinfected before the threshold is reached, the worm cannot spread in the Internet. This strategy can effectively contain the spread of uniform scanning worms in local networks. When the *global* total number of infected hosts is low, the number of infected hosts in any local networks must also be very low.

Our worm containment scheme can be deployed at end hosts and edge routers. When it is deployed at routers, special care must be taken when the network uses proxies or NAT. The counting of distinct destination IPs for each host is best placed at the proxies or inside the NAT. When the worm containment system is deployed outside NAT and proxies, a higher limit on M_C can be allocated based on the normal traffic patterns. Deploying our scheme at routers also requires protection against spoofing of IP addresses in order to correctly identify the offending hosts. The host's MAC address and IP address can be used together for identifying the offending host. Commercial products, such as *Counter-Point* by Mirage Networks, uses a host's MAC address for

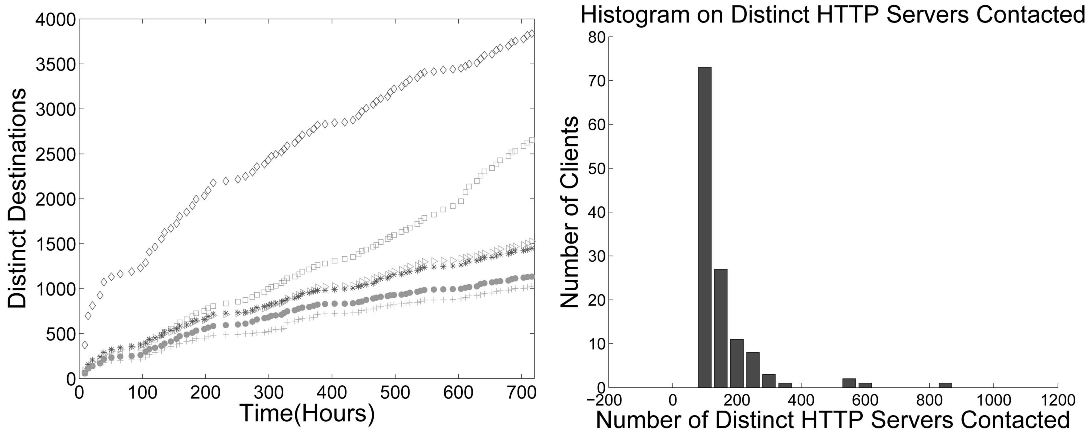


Fig. 4. (a) Number of distinct IPs contacted over 30 days for the six most active hosts (LBL trace data). (b) Histogram of distinct HTTP Servers contacted over one week, excluding the three proxies and two other active hosts with 5K to 6K destinations (Bell Lab trace data).

identifying an infected host. The MAC addresses can be made immune to spoofing by configuring the port on the layer 2 switch as a secure port [5]. This functionality is available in most common switches, such as the Cisco Catalyst 6500 Series where it is disabled by default [3]. The port security feature allows one to restrict input to an interface by limiting and identifying MAC addresses of the workstations that are allowed to access the port. When one assigns secure MAC addresses to a secure port, the port does not forward packets with source addresses outside the group of defined addresses.

In the future, we would like to design and implement our worm containment module at the edge routers and local routers. We plan to use real trace data to test the performance, usability, and scalability of our scheme. Meanwhile, we are also interested in evaluating the deployment of our scheme at strategically positioned routers. Defending against worms that exploit high vulnerability density is another interesting topic for future research.

5 Local Preference Scan Worms

Local preference scanning (LPS) worms, such as Code Red II, scan more intensely in local networks. Code Red II scans a completely random address only 1/8 of the time. It scans the same /16 network 1/2 of the time, and it scans the same /8 network 3/8 of the time. When vulnerable hosts are more dense in the local networks, the LPS worms spread much faster in the enterprise network. The faster spread is the result of the following two factors. One is that LPS worms scan the local network thousands of times more than do uniform scanning worms. The other factor is the higher vulnerability density in the local network due to similar configurations among machines in a subnet.

In this section, we extend our worm containment system developed in Section 4 to contain LPS worms in enterprise networks by removing the infected hosts in a timely manner. In order to prevent DOS attacks using IP spoofing, we need to use the host's MAC address and IP address to identify the offending host before it is taken offline, or use antispoofing software with our scheme. We show that our LPS containment scheme prevents the worm from spreading inside the local networks. Moreover, we provide analysis and simulations of global containment of LPS worms. When the LPS worm containment scheme is deployed in an enterprise network with traditional firewalls around the enterprise network boundary, worm containment inside the local network can be achieved without the requirement of participation and coordination from outside networks. Our simulations show that when this scheme is partially deployed, the LPS worm is likely to be contained on a global scale when the initially infected hosts are in the protected networks. Our analysis and simulation shows that when this scheme is deployed 100 percent, we can achieve not only local containment but also global containment.

Our approach relies on the fundamental argument that there naturally exist large swaths of unused IP address space in today's IPv4 infrastructure (roughly 25 percent of the address space is used). The legitimate hosts are unlikely to scan multiple times to the unused addresses and, therefore, the number of such scans can be used to estimate the number of *worm* scans. When the number of worm scans exceeds the limit due to the epidemic theory, the machine is quarantined to prevent the spread of the worm.

This approach is also used in LaBrea [19], DIB/TRAN [20], and the worm monitoring system proposed by Zou et al. [32]. Network Intrusion Detection System Bro [26] can also be configured to perform dark-address detection. Commercial worm containment systems such as *CounterPoint* by Mirage Networks [21] and CounterACT by Forescout [13] also use the dark-address scan detection approach. In the dark-address detection approach, a worm infection is declared when a host has attempted to connect to the dark addresses a number of times (denoted by N).

It is generally recognized that the choice of N is important —too small an N value will result in high number of false alarms, whereas too large an N value will result in a detection delay, allowing the worms to spread. Our contribution is that we provide an upper bound on N to guarantee worm containment. N is given in terms of the local vulnerability density p_1 and the dark-address density u. To contain worms, we require $N \leq \frac{u}{p_1}$.

5.1 Model

As in our model for uniform scanning worms, we take a host-centric view of worm propagation. We consider an infected host along with the /16 network and the /8 network to which it belongs. Let p_1, p_2, and p_3 be the vulnerability densities of the /16 network, /8 network, and the Internet, respectively. Let M_1, M_2, and M_3 be the total number of scans by an infected host in the /16 network, /8 network and the Internet. Let f_1, f_2, and f_3 be the fraction of times a worm scans the /16 network, the /8 network, and the Internet. The total number of scans is $M = M_1 + M_2 + M_3$ and $M_i = f_i M$ for $i = 1, 2, 3$.

Let ξ_1, ξ_2, and ξ_3 be the number of offsprings an infected host produces in the /16 network, the /8 network, and the Internet when it scans M times. The random variables ξ_i are binomially distributed with parameters (M_i, p_i), $i = 1, 2, 3$. Moreover, $E[\xi_i] = M_i p_i = M f_i p_i$. The total number of offsprings ξ produced by an infected host is then $\xi = \xi_1 + \xi_2 + \xi_3$, and $E[\xi] = M(f_1 p_1 + f_2 p_2 + f_3 p_3)$. When the vulnerable hosts are much denser inside the internal networks (that is, $p_1 >> p_3$ and/or $p_2 >> p_3$), LPS worms pose a more serious threat to the internal networks than to the rest of the Internet. This is because a single infected host produces far more offsprings inside local networks.

Our worm containment system for uniform scanning worms restricts the total number of outgoing connections to below $M \leq 1/p$. If we adopt the same scheme to contain LPS worms in an enterprise network consisting of several /16 networks, in each /16 network, M_1 needs to be considerably smaller because of the higher vulnerability density p_1 in the network. In a /16 network with 600 vulnerable hosts ($p_1 \approx 0.01$), the original containment scheme requires $M_1 \leq 100$, that is, no host is allowed to contact more than 100 distinct destinations in that /16 network. It is overrestrictive because some benign hosts may in fact contact 100 distinct IP addresses in the local network over a period of time.

In the following, we will show how to extend our worm containment system to contain LPS worms in enterprise networks. We use a /16 network to illustrate our new scheme.

5.2 LPS Worm Containment System Analysis

Our LPS worm containment system limits the total scans a host can make to the dark-address space. In a containment cycle (weeks or months), if a host scans the dark addresses N times, it is automatically disconnected from the network for a thorough checkup. It is desirable that N be large so that normal activities will not be disrupted while still containing LPS worms effectively.

If a host is infected, its total number of scans to dark addresses N provides essential information on the total number of worm scans $M_{1,w}$ and the number of offsprings ξ_1 it produces. To prevent the worms from spreading, we must limit the expected number of offsprings to below 1. This can be achieved by limiting N, the total number of dark-address scans per host. We provide the probability distribution of $M_{1,w}$ and ξ_1 for a given value of N in the following proposition.

Proposition 2. *Let u be the density of the dark IP addresses and p_1 be the vulnerable density in an enterprise network $(u + p_1 \leq 1)$. Suppose an infected host is removed from the network when it hits the dark address the Nth time. Let $M_{1,w}$ be the total worm scans in the local network by an infected host and ξ_1 be the number of offsprings produced by a single infected host in the network. Then,*

1. *$M_{1,w}$ is a negative binomial random variable having distribution*

$$P\{M_{1,w} = m\} = \binom{m-1}{N-1} u^N (1-u)^{m-N}, \quad m = N, N+1, N+2, \cdots,$$

and $E[M_{1,w}] = N/u$.

2. *ξ_1 is a negative binomial random variable having distribution*

$$P\{\xi_1 = k\} = \binom{k+N-1}{k} \left(\frac{p_1}{p_1+u}\right)^k \left(\frac{u}{p_1+u}\right)^N, \quad k = 0, 1, 2, \cdots,$$

and $E[\xi_1] = Np_1/u$.

Proof.

1. Suppose an infected host scans m times and is stopped at the Nth scan to the unused IP addresses, $(m \geq N)$. The mth scan falls into the unused IP addresses. Among the first $m-1$ scans, it scans the unused IP addresses $N-1$ times and scans the vulnerable and healthy hosts $m-N$ times. Thus,

$$P\{M_{1,w} = m\} = \binom{m-1}{N-1} u^N (1-u)^{m-N}, \quad m = N, N+1, \cdots.$$

Therefore, $M_{1,w}$ is a negative binomial random variable with mean $E[M_{1,w}] = N/u$.

2. When the infected host is removed after N scans to the unused IP addresses

$$P\{\xi_1 = k | M_{1,w} = m\} = \binom{m-N}{k} \left(\frac{p_1}{1-u}\right)^k \left(1 - \frac{p_1}{1-u}\right)^{m-N-k}.$$

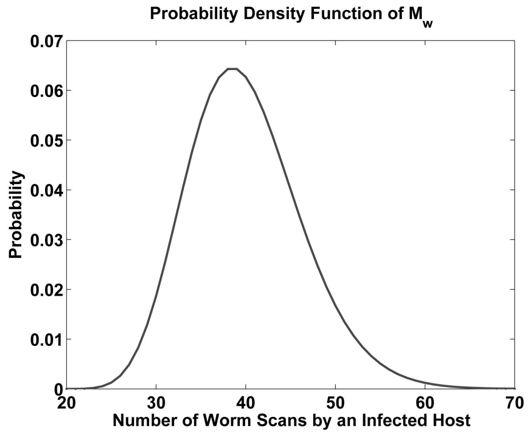


Fig. 5. Probability Density Function of $M_{1,w}$ $(\mathrm{u} = 0.5, \mathrm{N} = 20)$.

Hence,

$$\begin{aligned}P(\xi_1 = k) &= \sum_{m=N+k}^{\infty} P(\xi_1 = k | M_{1,w} = m) P(M_{1,w} = m) \\ &= \sum_{m=N+k}^{\infty} \binom{m-N}{k} \left(\frac{p_1}{1-u}\right)^k \left(\frac{1-u-p_1}{1-u}\right)^{m-N-k} \\ &\quad \times \binom{m-1}{N-1} u^N (1-u)^{m-N} \\ &= \sum_{m=N+k}^{\infty} \binom{m-N}{k} \binom{m-1}{N-1} p_1^k (1-u-p_1)^{m-N-k} u^N \\ &= \binom{k+N-1}{k} p_1^k u^N \\ &\quad \times \sum_{m=N+k}^{\infty} \binom{m-1}{k+N-1} (1-u-p_1)^{m-N-k} \\ &= \binom{k+N-1}{k} p_1^k u^N (p_1+u)^{-N-k} \\ &= \binom{k+N-1}{N-1} \left(\frac{p_1}{p_1+u}\right)^k \left(\frac{u}{p_1+u}\right)^N.\end{aligned}$$

Thus, ξ_1 is a negative binomial random variable with parameters $(N, \frac{u}{p_1+u})$, and $E[\xi_1] = \frac{Np_1}{u}$. □

Proposition 2 shows that the number of worm scans $M_{1,w}$ is not affected by p_1, but the number of internal offsprings ξ_1 is a function of u, N, and p_1. Fig. 5a shows the pmf of $M_{1,w}$ for $u = 0.5$ and $N = 20$. Fig. 6a illustrates the pmf's of ξ_1 for the same u and N values as in Fig. 5a, with p_1 values 0.01, 0.03, and 0.05. In particular, when $p_1 = 0.05$, $E[\xi_1] = 2$. That is, if $u = 50\%$, $N = 20$, and there are roughly 3,000 vulnerable hosts in a /16 network, one infected host on the average infects two vulnerable hosts during the initial worm outbreak.

To prevent the worms from spreading, we will limit the expected number of offsprings per host $E[\xi_1]$ to below 1 by the extinction theorem for branching processes. Since $E[\xi_1] = Np_1/u$, we require $N \leq u/p_1$, we denote $N_{max} = \lfloor u/p_1 \rfloor$.

The local vulnerability density p_1 will decrease when more vulnerable hosts become infected. The above bound

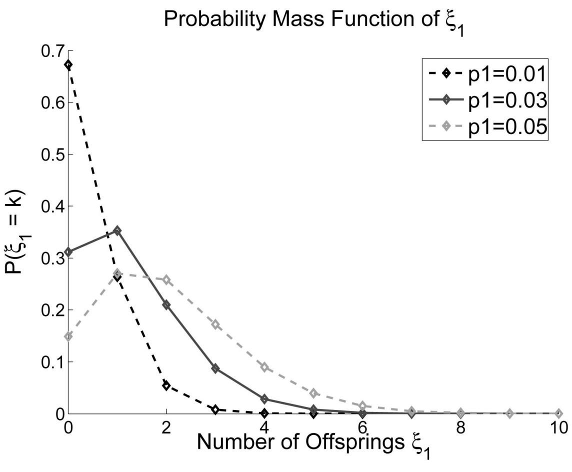


Fig. 6. Probability Density Functions of ξ_1 $(\mathrm{u} = 0.5, \mathrm{N} = 20)$.

on N is derived using the vulnerability density in the early stage when it is the largest, so the bound N_{max} is still valid when more hosts become infected. Moreover, when this bound is enforced, the total number of the infected hosts would be small, and a large fraction of vulnerable hosts is automatically protected from the LPS worms.

Our next proposition provides analysis on the global impact of the LPS worm containment.

Proposition 3. *Let p_1, p_2, p_3 be the vulnerability densities for all /16 networks, /8 networks, and in the whole Internet, respectively. The LPS worm scans /16 networks, /8 networks, and the Internet f_1, f_2, and f_3 fraction of times such that $f_1 + f_2 + f_3 = 1$. Let ξ be the number of offsprings an infected host produces. Then, $E[\xi] = \frac{N}{u}(p_1 + p_2\frac{f_2}{f_1} + p_3\frac{f_3}{f_1})$.*

Proof. Let ξ_2 be the number of offsprings an infected host produces in the /8 network but outside the /16 network. When the infected host scans the /16 network $M_{1,w}$ times, it scans /8 networks it belongs to $M_{1,w} \cdot (f_2/f_1)$ times

$$E[\xi_2] = p_2 E[\text{number of scans in/8 network}] = p_2 E\left[M_{1,w} \cdot \frac{f_2}{f_1}\right]$$
$$= p_2\frac{N}{u} \cdot \frac{f_2}{f_1} = \frac{Np_2}{u} \cdot \frac{f_2}{f_1}.$$

Similarly, the expected number of offsprings an infected host produces outside the /8 network is $E[\xi_3] = \frac{Np_3}{u} \cdot \frac{f_3}{f_1}$.

Therefore, the expected number of offsprings each infected host produces is

$$E[\xi] = E[\xi_1 + \xi_2 + \xi_3]$$
$$= \frac{Np_1}{u} + \frac{Np_2}{u} \cdot \frac{f_2}{f_1} + \frac{Np_3}{u} \cdot \frac{f_3}{f_1} = \frac{N}{u}\left(p_1 + p_2\frac{f_2}{f_1} + p_3\frac{f_3}{f_1}\right).$$

□

When $E[\xi] < 1$, the LPS worm can be contained globally, and the average global total number of infected hosts is $E[I] = I_0/(1 - E[\xi])$. Proposition 3 also tells us that our LPS worm containment system cannot contain uniform scanning worms, because their scanning pattern has the following parameters: $f_2/f_1 = 256$ and $f_3/f_1 = 65,536$. This will result in a large number of infections outside the local networks. Therefore, we need both containment schemes to contain all scanning worms.

The scanning strategy of Code Red II has the following parameters: $f_1 = 1/2$, $f_2 = 3/8$, and $f_3 = 1/8$. Thus, $f_2/f_1 = 3/4$, $f_3/f_1 = 1/4$. Assume our networks have the following parameters: $p_1 = 0.02$, $p_2 = 0.01$, and $p_3 = 10^{-4}$. If we use $N = 20$, $u = 0.5$ for LPS containment, then $E[\xi] = 40$ $(0.02 + 0.01 \cdot 0.75 + 10^{-4} \cdot 0.25) = 1.101$. In this scenario, the worm will spread to the Internet. If the LPS containment system uses $N = 10, u = 0.5$, then the worm will be contained globally since $E[\xi] = 0.55 < 1$.

5.3 LPS Worm Containment System Deployment

The above analysis suggests that in order to protect local networks from worm infections, the local network addresses should be allocated in such a way that the ratio $\frac{u}{p_1}$ is larger than the number of dark-address connections expected from a normal host (denoted as N_t). Choosing N so that $N_t < N < \frac{u}{p_1}$ allows worm containment without intruding on normal-user activities.

To deploy the new worm containment system in an existing network, we need to know the parameters u, p_1, and N_t. Here, u is the density of dark-address space; p_1 is the estimated vulnerability density inside the network; N_t is a tolerable threshold value—a benign host will not mistakenly scan the dark addresses more than this value in a containment cycle. N_t is an optional parameter that may be set by the system administrator. The value of the local vulnerability density p_1 can be estimated based on the most common applications with open server ports. The value of u is known to the network administrator from the number of machines in the local network. Using u and p_1, we can calculate $N_{max} = \lfloor(\frac{u}{p_1})\rfloor$. The actual threshold N for the containment system should be less than N_{max} to guarantee containment and greater than N_t to not interfere with normal user activities. When $N_{max} \geq N_t$, we can set $N = N_t$. However, when $N_{max} < N_t$, we could set $N = N_{max}$ and tolerate a higher rate of false alarms in order to contain the worms.

Our analysis shows that if a local network has very high vulnerability density, the dark address-based scheme will be too restrictive in the number of dark-address space scans allowed. One solution for the high vulnerability density networks is to adopt the 24-bit block private address scheme. For example, networks utilizing private addresses 10.0.0.0/24 can accommodate 360,000 hosts with $u \approx 0.98$. If we make a conservative estimate on p_1 by assuming all these hosts are vulnerable, $p_1 \approx 0.02$. The value of $N_{max} = 49$.

We agree that an organization that has u/p_1 close to 1 and is unwilling to take measures (such as private address space) to increase the value, our scheme will not be useful. In general, local networks with dense and homogeneous software deployment are at a higher risk of worm infections. However, our simulation of incremental deployment shows that our scheme is beneficial even for those unprotected networks when the initial infection starts from the protected networks.

The goal of our worm containment system is to probabilistically contain the worms in a nonintrusive manner without explicitly detecting the infected hosts. Our scheme allows hosts to stay in the infected state for some time until their dark-address scans reach the limit, and it still guarantees containment. This is fundamentally different from worm detections.

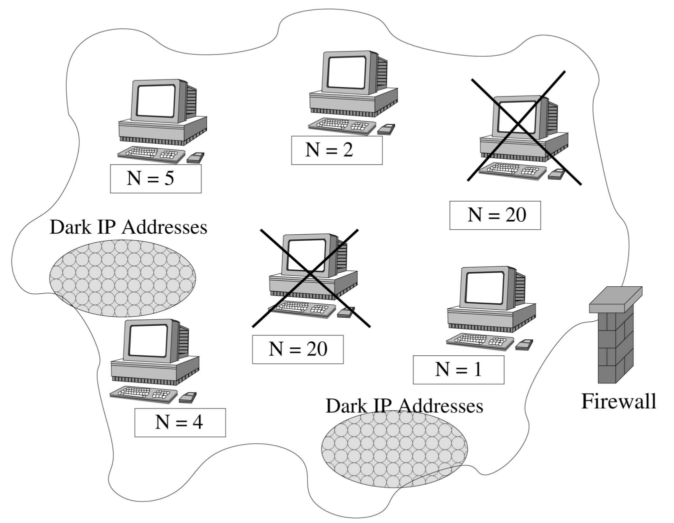


Fig. 7. Worm containment systems for the preference scan worms. The value N_{max} is chosen to be 20. Two hosts are disconnected from the network because their total dark-address space scans has reached 20.

By combining traditional firewalls around the enterprise network boundary and our worm containment system within the enterprise network, worm containment can be achieved without the requirement of participation and coordination from outside networks. This allows for incremental deployment of the worm containment system for individual networks. Fig. 7 illustrates the worm containment system for preference scanning worms.

A strong firewall at the enterprise network boundary is necessary to achieve worm containment inside the enterprise network. Consider an enterprise network consisting of a /16 network. Consider the situation when worms are spreading outside the network with more than 65K hosts already infected. One random scan from each infected host to the enterprise network will cause a significant fraction of vulnerable hosts to be infected. The first approach to counter the problem is to apply the same limit on the number of scans to unused address space and to an external host as to an internal host. When the external host reaches this limit, a firewall rule is introduced to block any incoming scans from the suspected host. The challenges in creating accurate firewall rules (countering address spoofing, for example) are out of the scope of this paper, and the reader is referred to [43, pp. 340-351] for details. However, this approach fails if this suspected external host is not prevented from infecting other vulnerable external hosts. Failing that, more infected hosts are generated in the external network, which in turn scan to the internal network. Repeating the process of blacklisting the external hosts leads to the situation where all susceptible internal hosts are eventually infected. Since the spread of worms outside the networks cannot be controlled by an internal worm containment system, it suggests a complementary approach. The approach is that we restrict the number of hosts (servers) that are accessible from outside the enterprise network. All other hosts should be hidden behind the firewall and not accessible from outside networks. In the event of Internet scale worm outbreaks, the small number of externally accessible servers may be infected. However, these servers cannot spread the worms to the internal networks because the worm containment system will disconnect the servers when they become infected. Moreover, the threshold value N for the servers can be carefully tuned down to a smaller number to further restrict the worm propagation. The subsetting to create a few externally accessible servers seems feasible in many enterprise settings since only a few servers (for example, Web server) need be exposed to the external network.

It is challenging to contain hitlist worms. Hitlist worms create a target list of probable victims. However, as noted in [44], building the hitlist is nontrivial. A small hitlist can be compiled from readily accessible public sources, such as a metaserver that maintains a list of servers for different games (GameSpy is one example). Such a hitlist may accelerate a scanning worm but is unlikely to be very damaging by itself. Building comprehensive hitlists takes more effort. A distributed scan to find vulnerable hosts appears a likely strategy. The scans must be low rate to avoid detection. Since our worm containment system can contain the slow spreading worms, it will prevent the hitlist from being built through such low-rate random scanning.

6 Simulation Results

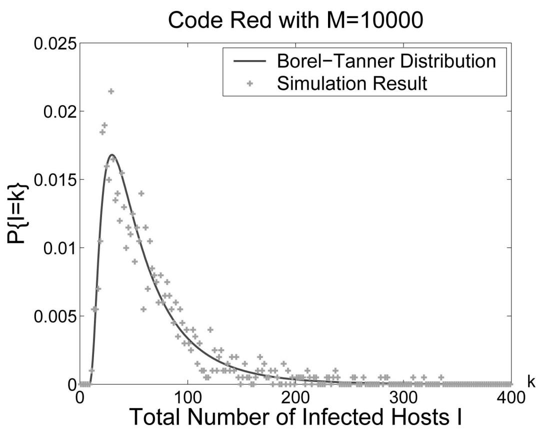
We use a discrete event simulator to simulate scanning worm propagation with our defense strategies. We first simulate a *uniform scanning worm* with our containment scheme. In this simulator, each vulnerable host is assigned an IPv4 address randomly, and each will be in one of the three states: *susceptible*, *infected*, and *removed*. A host is in a *removed* state if it has sent M scans. The *infected* hosts independently generate random IP addresses to find the victims. If the random IP address matches any of the IP addresses of the hosts in the *susceptible* state, the susceptible host will become infected.

In our simulation for Code Red, we used $V = 360,000$ for the vulnerable population size and $I_0 = 10$ for the number of initially infected hosts. We used $M = 10,000$, which is below the threshold required for worm extinction. In this case, $p = 8.38 \times 10^{-5}$ and $\lambda = Mp = 0.838$.

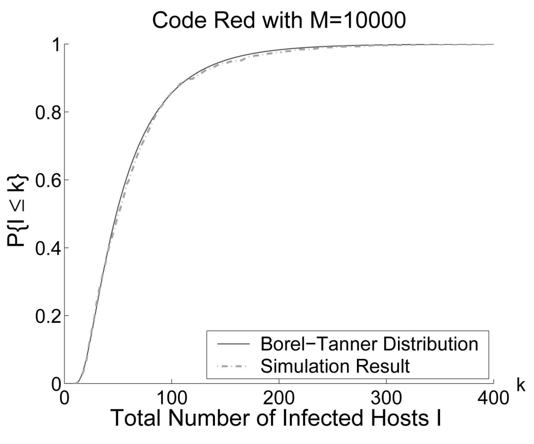
As discussed earlier, the total number of infected hosts I measures how well a worm is contained. We ran this simulation 1,000 times and collected the values of I. Fig. 8a shows the relative frequency of I from our simulations and the probability density function of I obtained from our theoretical results in Section 3. Fig. 8b shows the relative *cumulative* frequency of I from our simulations and the *cumulative distribution function* of I from the theoretical analysis.

The simulation results validate the accuracy of our model and the effectiveness of our containment strategy. Figs. 8a and 8b demonstrate that our simulation results match closely with the theoretical results from Section 3. We can see in Fig. 8b that with high probability (0.95), the total number of infected hosts is held below 150 hosts. As mentioned in Section 3, one can reduce the spread of infection by further reducing the value of M.

To demonstrate the variability of worm propagation, we show two sample paths among our 1,000 simulation runs in Fig. 9. In one scenario depicted in Fig. 9a, there are a total of approximately 300 hosts infected. Fig. 9b shows another scenario when there are 55 total infected hosts. In the first scenario, the active number of infected hosts (number infected—number removed) is held below 30 at all times. This is due to our countermeasure that when a host scans $M = 10,000$ times, it is removed. The worm ceased spreading after all infected hosts were removed. In the second scenario, the removal process quickly catches the infection process, so that the worm dies out rapidly. We used $M = 10,000$ for both

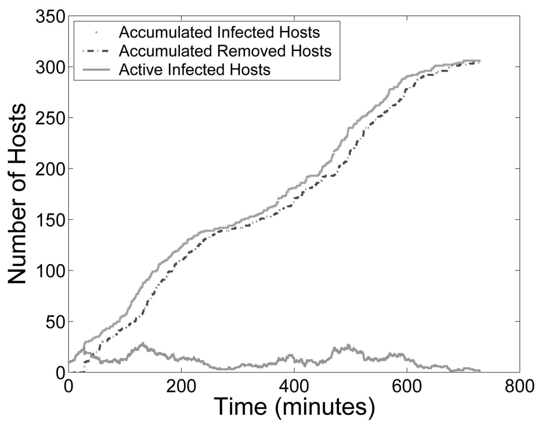


(a)

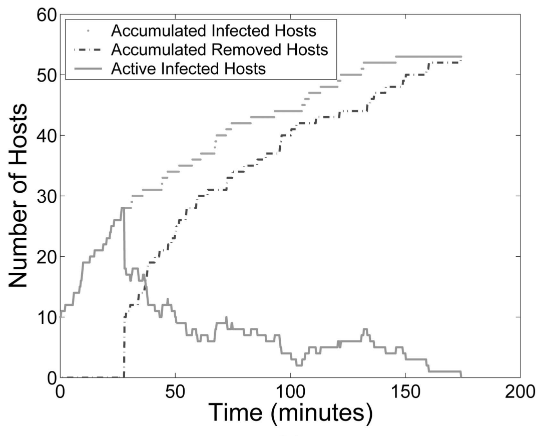


(b)

Fig. 8. Probability density and cumulative distribution of I.



(a)



(b)

Fig. 9. Variability of worm propagation: two simulation runs with identical parameters (Code Red).

scenarios. Using formula provided in Section 3, we obtain $E(I) = 58$ and $\sigma(I) = 45$. The variation is large and can be significant in the early stage, which will have significant effect in modeling the latter stage of growth of the worm. Stochastic models need to be used to capture this variation.

We used a scan rate of 6 scans/second for Code Red for the purpose of illustrating worm propagation and containment with respect to time. This scan rate is taken from the empirical study in [22]. Fig. 9a shows that the worm dies out completely in 750 minutes. In the simulation run shown in Fig. 9b, the worm dies out in less than 200 minutes.

We also ran our simulations with SQL Slammer parameters. Here, we used $\mathrm{V} = 120,000$ as used in [20], $I_0 = 10$, and $M = 10,000$. The experiments with Slammer show a close match between predicted and observed values, and this is borne out in Fig. 10. The worm containment scheme contains the infection to below 20 hosts (that is, only 10 newly infected hosts) with a very high probability.

To illustrate the effectiveness of our LPS worms, we also simulated our LPS worm containment system in a /16 network. In this simulator, the addresses for the vulnerable hosts and the dark IP addresses are randomly chosen from the /16 network.

The *infected* hosts independently generate random /16 IP addresses to find victims. If the random IP address matches any of the IP addresses of the hosts in the *susceptible* state, the susceptible host will become infected. If it matches any of the dark IP addresses, the counter for its dark-address scans will be increased. An infected host is *removed* if it has sent N scans to the dark addresses.

In the first three scenarios for the LPS worm containment system, we set the unused IP address density to equal 50 percent ($u = 0.5$), as in [20], and the number of initial infections I_0 is set to 20. We ran each of the scenarios 1,000 times and collected the number of total infections in the /16 network for each run.

Fig. 11a shows the effectiveness of the worm containment scheme when $N = 20$ for various vulnerability densities. As shown in Fig. 11a, with high probability (0.99), the total number of infected hosts in the network is less than 45 when there are 300 and 600 vulnerable hosts. When number of vulnerable hosts is increased to 1,200, the containment system is less effective due to increased vulnerability density.

When the number of vulnerable hosts is 1,800, having the same parameters as in the above scenarios ($u = 0.5$ and $N = 20$) will not contain the worms. Fig. 11b shows the cumulative distribution function of the total number of infections I ($u = 0.5$, $N = 20$, and $I_0 = 20$). As shown in Fig. 11b, with probability more than 0.9, the total number of infections will exceed 500. This is because, in this case, $p_1 = 0.0275$ so that $N_{max} = \lfloor 0.5/0.0275 \rfloor = 18$. When we set $N = 20$ for the worm containment systems, the average number of offsprings from an infected host during the early phase is approximately 1.1, only slightly higher than 1. However, this

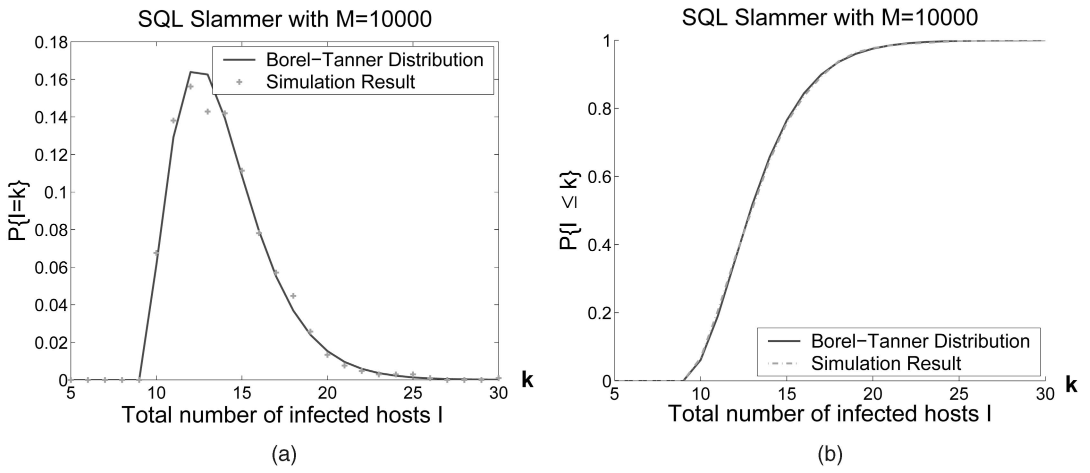


Fig. 10. Probability density and cumulative distribution of I (SQL slammer).

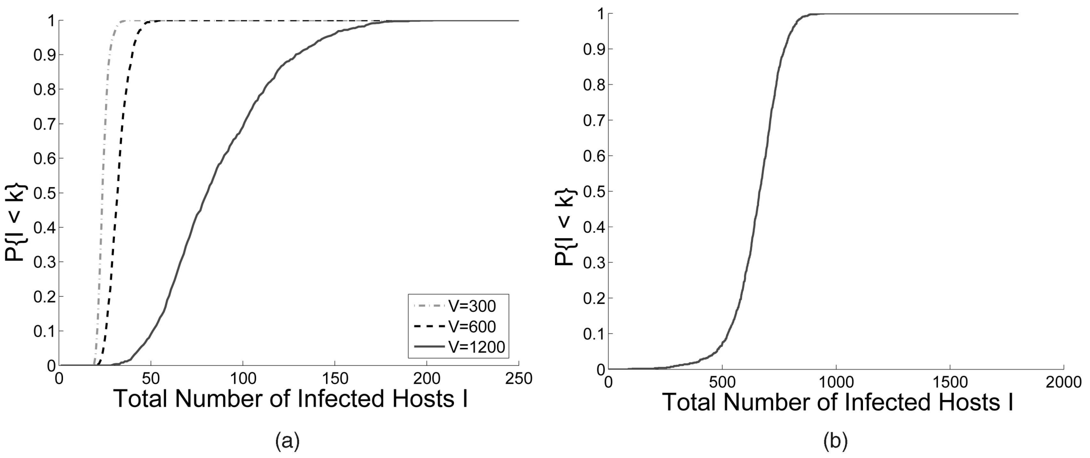


Fig. 11. The cumulative distribution of total number of infected hosts ($u = 0.5$, $N = 20$, and $I_0 = 20$). In (a), $V = 300, 600, 1,200$, and in (b), $V = 1,800$.

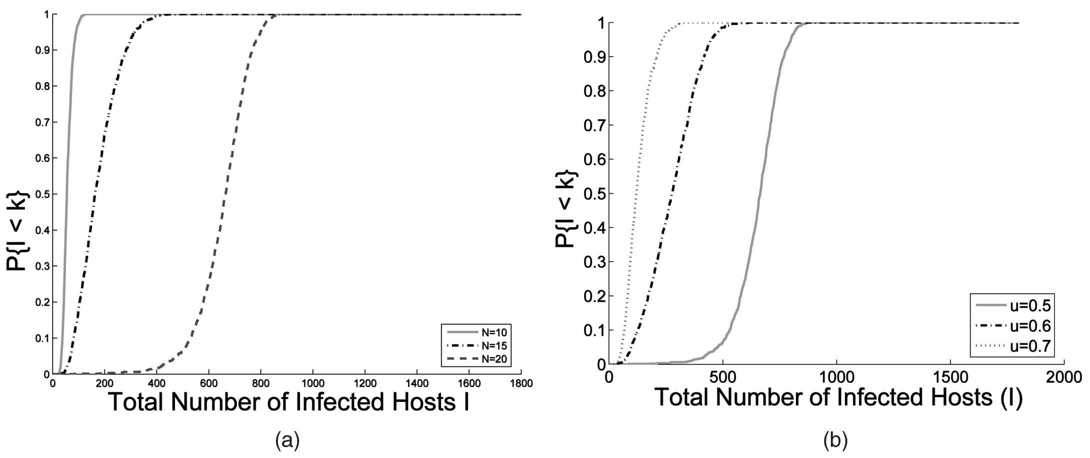


Fig. 12. The cumulative distribution of total number of infected hosts ($V = 1800$, $I_0 = 20$). In (a), $u = 0.5$, N varies; in (b), $N = 20$, u varies.

results in a large fraction of vulnerable hosts getting infected. Because the local vulnerability density p_1 decreases as more vulnerable hosts become infected, the number of offsprings from later infected hosts will decrease. Therefore, not all the vulnerables in the local network will be infected. The total number of the infected hosts is less than 1,000 (out of 1,800) with high probability (Fig. 12).

Our analysis gives a network designer an easily quantifiable trade-off. First, reduce N to give tighter probabilistic bound on the total number of infected hosts at the risk of higher false alarm. Second, increase dark-address space to again give the tighter bound at the risk of not fully utilizing the allocated IP address space.

Next, we simulated incremental deployment scenarios on a single /8 network that contains 256 /16 networks. We simulated an LPS worm that randomly scans the local /16 network 1/2 of the time and the /8 network 1/2 of the time. Each /16 network has V $= 1,200$ vulnerable hosts.

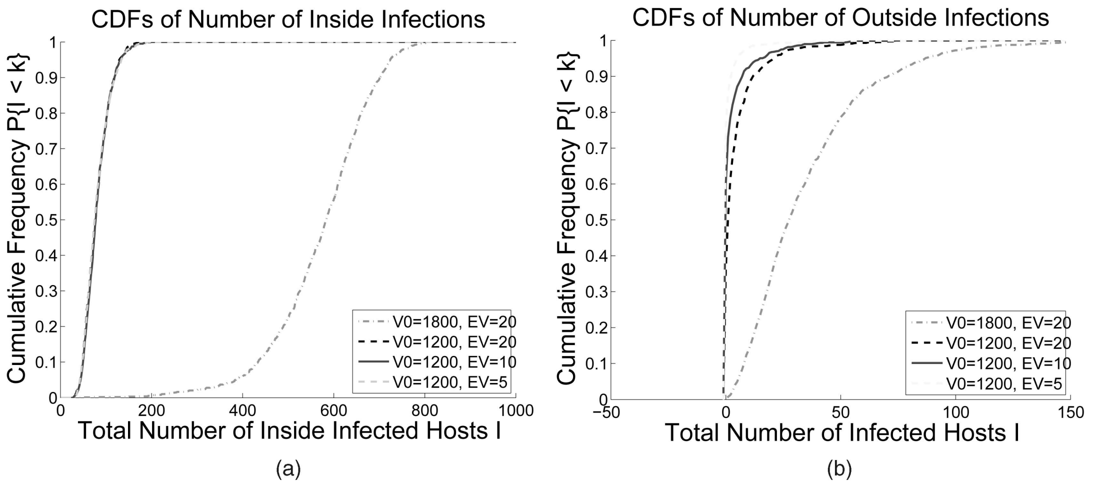


Fig. 13. Full Deployment of LPS containment scheme. (a) Infections inside the original infections network. (b) Infections outside the original infected networks.

In our simulations with 100 percent deployment (that is, all the /16 networks within the /8 have the defense mechanism deployed), there are 5 to 20 of these vulnerable hosts that are externally visible (denoted as EV in the plot). Only these externally visible hosts can be infected by hosts scanning from other networks. In one scenario, the network with initially infected hosts has $V_0 = 1,800$ vulnerable hosts. In three other scenarios, the network with initially infected hosts has $V_0 = 1,200$ vulnerable hosts. In all the scenarios, the initially infected hosts lie within a single /16 network and are 20 in number.

Our results are shown in Fig. 13. We can see that when $V_0 = 1,200$, the number of infections in the local network is between 50 to 150, and the total number of infections in all outside networks is very small. When $V_0 = 1,800$, the total number of infections outside of the network is also small (about 100), despite a large number of hosts being infected inside the network. Our analysis and simulations both show that our LPS worm containment scheme can indeed provide global containment with universal deployment.

Next, we simulated partial deployment scenarios with only one protected network, as well as 50 percent, 75 percent, and 90 percent of the /16 networks being protected by our LPS worm containment scheme. There are 20 externally visible hosts ($EV = 20$) in each of the /16 networks. We only simulate the scenarios when the initially infected hosts are in protected networks. When the initially infected host reside in an unprotected network, the worm will spread to all the unprotected networks eventually. The number of the initially infected hosts is 5 ($I_0 = 5$) in our simulations.

We ran our simulation 1,000 times in each partial deployment scenario. Our results are summarized in Table 2. In all scenarios, the worm is completely contained within the originating network over 76 percent of the time. When only 50 percent of the networks are protected, the worm escapes to the unprotected networks only 12.7 percent of the time; it is contained in other protected networks 10.9 percent of the time and is completely contained in the originating network 76.4 percent of the time. When the LPS worm containment system is deployed in only one network in which the initially infected host originates, there is also a 76.9 percent chance that the worm is completely contained in the local network. This is not surprising since the protected networks have the same configurations, and the probability that the infection will spread outside the network is the same. However, the more networks deploy the LPS worm containment system, the less likely the worm infection will spread to the unprotected networks. Higher participation in LPS worm containment also makes it less likely for the initial infection to start in the unprotected networks.

The reason is that the average total internal infections is about 19, starting with five infected hosts. Average scans per infected host going outside the /16 network is about 40 before it gets quarantined. Therefore, the total number of scans from the originating /16 network on an average is less than 800. The vulnerability density in the /8 network $p_2 = 20/65536 = 3 \times 10^{-4}$ when $\mathrm{EV} = 20$, assuming pessimistically that all the external visible hosts are vulnerable. The expected number of external hosts that would be infected by the originating /16 network is very small (about 0.23).

Our simulation result shows that partial deployment of our LPS worm containment scheme can benefit the global network. If the initially infected host is in an unprotected network, the worm spread will be slowed down, depending on the how large the deployment base is. Let V_u be the total number of vulnerable hosts from an unprotected network, and $p_u = V_u/2^{32}$. If the infection originates from the unprotected networks, the worm propagation is at least the speed as if the vulnerability density is $p = p_u$.

Our analysis and simulation shows that our LPS worm containment system is very effective when there is a 100 percent deployment. When there is only a partial deployment, it protects the local networks and provides global benefit. Since this scheme benefits the local network, it is likely to be adopted by more organizations over time.

TABLE 2
Simulation Results on Partial Deployment

Networks with LPS Worm Containment	90%	75%	50%	ONE
Contained in Originating Network	76.6%	77.6%	76.4%	76.9%
Contained in Protected Networks	20%	15.2%	10.9%	0
Escaped to Unprotected Networks	3.4%	7.2%	12.7%	23.1%

7 Conclusion

In this paper, we have studied the problem of combating Internet worms. To that end, we have developed a branching process model to characterize the propagation of Internet worms. Unlike deterministic epidemic models studied in the literature, this model allows us to characterize the early phase of worm propagation. Using the branching process model, we are able to provide a precise bound M on the total number of scans that ensure that the worm will eventually die out. Further, from our model, we also obtain the probability that the total number of hosts that the worm infects is below a certain level, as a function of the scan limit M. The insights gained from analyzing this model also allow us to develop an effective and automatic worm containment strategy that does not let the worm propagate beyond the early stages of infection. Our strategy can effectively contain both fast scan worms and slow scan worms without knowing the worm signature in advance or needing to explicitly detect the worm. We show via simulations and real trace data that the containment strategy is both effective and nonintrusive.

We extended the worm containment scheme to local preference scanning worms. In this scheme, we restrict the total number of scans per host to the dark-address space. We derive the precise bound N on the total number of scans to the dark-address space, which ensures that the worm will be contained. This containment scheme, combined with firewalls at the network boundary, allows for incremental deployment of the worm containment system without participation of outside networks.

For further work, we would like to propose a statistical model for the spread of topology-aware worms and subsequently design mechanisms for automatic containment of such worms. We would also like to characterize the deviation of our proposed branching process model from the "ideal" stochastic epidemic model, assuming that the values of its rich set of parameters were available. Finally, we would like to port our worm containment schemes to edge routers and local routers and to evaluate the performance using real data from enterprise networks.

Acknowledgments

The authors would like to thank the anonymous reviewers for their insightful feedback, which greatly improved the quality of this paper. They also thank Professor William Cleveland and Joerg Micheel for their tremendous help with the trace data from Bell Labs. This work is partially supported by the US National Science Foundation grant 0335247-ANI, 0523249-CNS, and a US NSF Graduate Fellowship.

References

[1] CAIDA, "CAIDA Analysis of Code-Red," http://www.caida.org/analysis/security/code-red/, 2007.
[2] "The Cost of Code Red: $1.2 Billion," *USA Today News*, http://www.usatoday.com/tech/news/2001-08-01-code-red-costs.htm, 2001.
[3] Cisco Documentation, "Configuring Port Security," http://www.cisco.com/univercd/cc/td/doc/product/lan/cat6000/12_1e/swconfig/port_sec.htm, 2007.
[4] H. Andersson and T. Britton, "Stochastic Epidemic Models and Their Statistical Analysis," *Lecture Notes in Statistics*, vol. 151, 2000.
[5] J. Bartiomiejczyk and M. Phipps, *Preventing Layer 2 Security Threats*, http://searchnetworking.techtarget.com/tip/0,289483,sid7_gci1009100,00.html, 2007.
[6] V.H. Berk, R.S. Gray, and G. Bakos, "Using Sensor Networks and Data Fusion for Early Detection of Active Worms," *Proc. SPIE AeroSense*, vol. 5071, pp. 92-104, 2003.
[7] Z. Chen, L. Gao, and K. Kwiat, "Modeling the Spread of Active Worms," *Proc. IEEE INFOCOM '03*, pp. 1890-1900, 2003.
[8] P.C. Consul, "Generalized Poisson Distributions, Properties and Applications," *STATISTICS: Textbooks and Monographs*, vol. 99, Marcel Dekker, 1988.
[9] D.J. Daley and J. Gani, *Epidemic Modelling, An Introduction.* Cambridge Univ. Press, 1999.
[10] A. Ganesh, L. Massoulie, and D. Towsley, "The Effect of Network Topology on the Spread of Epidemics," *Proc. IEEE INFOCOM '05*, pp. 1455-1466, 2005.
[11] A. Ganesh, D. Gunawardena, P. Key, L. Massoulie, and J. Scott, "Efficient Quarantining of Scanning Worms: Optimal Detection and Coordination," *Proc. IEEE INFOCOM '06*, pp. 1-13, 2006.
[12] D. Dagon, X. Qin, G. Gu, W. Lee, J.B. Grizzard, J.G. Levine, and H.L. Owen, "HoneyStat: Local Worm Detection Using Honeypots," *Proc. RAID Symp.*, pp. 39-58, 2004.
[13] Foresount, *WormScout*, http://www.foresout.com/, Jan. 2008.
[14] J. Jung, V. Paxson, A.W. Berger, and H. Balakrishnan, "Fast Portscan Detection Using Sequential Hypothesis Testing," *Proc. IEEE Symp. Security and Privacy*, pp. 211-225, 2004.
[15] S. Karlin and H.M. Taylor, *A First Course in Stochastic Processes*, second ed. Academic Press, 1975.
[16] J.O. Kephart and S.R. White, "Directed-Graph Epidemiological Models of Computer Viruses," *Proc. IEEE Symp. Security and Privacy*, pp. 343-359, 1991.
[17] J.O. Kephart, D.M. Chess, and S.R. White, "Computers and Epidemiology," *IEEE Spectrum*, vol. 30, pp. 20-26, May 1993.
[18] J.O. Kephart and S.R. White, "Measuring and Modeling Computer Virus Prevalence," *Proc. IEEE Symp. Security and Privacy*, pp. 2-15, 1993.
[19] LaBrea, "LaBrea Technologies," http://www.labreatechnologies.com/, 2007.
[20] M. Liljenstam, D.M. Nicol, V.H. Berk, and R.S. Gray, "Simulating Realistic Network Worm Traffic for Worm Warning System Design and Testing," *Proc. ACM Workshop Rapid Malcode*, pp. 24-33, 2003.
[21] Mirage Networks, http://www.miragenetworks.com/, 2007.
[22] D. Moore, C. Shannon, and J. Brown, "Code-Red: A Case Study on the Spread and Victims of an Internet Worm," *Proc. ACM Internet Measurement Workshop*, pp. 273-284, 2002.
[23] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the Slammer Worm," *IEEE Security Privacy*, vol. 1, no. 4, pp. 33-39, July 2003.
[24] D. Moore, C. Shannon, G.M. Voelker, and S. Savage, "Internet Quarantine: Requirements for Containing Self-Propagating Code," *Proc. IEEE INFOCOM '03*, pp. 1901-1910, 2003.
[25] NLANR, "Bell Lab—I Data Set," http://pma.nlanr.net/Traces/long/bell1.html, 2007.
[26] V. Paxon, "Bro: A System for Detecting Network Intruders in Real-Time," *Computer Networks*, vol. 31, no. 23-24, pp. 2435-2463, Dec. 1999.
[27] S. Ross, *Stochastic Processes*, second ed. John Wiley & Sons, 1996.
[28] S. Staniford, V. Paxson, and N. Weaver, "How to Own the Internet in Your Spare Time," *Proc. Usenix Security Symp.*, pp. 149-167, 2002.
[29] M.M. Williamson, "Throttling Viruses: Restricting Propagation to Defeat Malicious Mobile Code," *Proc. IEEE Ann. Computer Security Applications Conf.*, pp. 61-68, 2002.
[30] C. Wong, C. Wang, D. Song, S. Bielski, and G.R. Ganger, "Dynamic Quarantine of Internet Worms," *Proc. IEEE Int'l Conf. Dependable Systems and Networks*, pp. 73-82, 2004.
[31] C.C. Zou, W. Gong, and D. Towsley, "Code Red Worm Propagation Modeling and Analysis," *Proc. ACM Conf. Computer and Comm. Security*, pp. 138-147, 2002.
[32] C.C. Zou, L. Gao, W. Gong, and D. Towsley, "Monitoring and Early Warning for Internet Worms," *Proc. ACM Conf. Computer and Comm. Security*, pp. 190-199, 2003.
[33] C.C. Zou, W. Gong, and D. Towsley, "Worm Propagation Modeling and Analysis under Dynamic Quarantine Defense," *Proc. ACM Workshop Rapid Malcode*, pp. 51-60, 2003.
[34] Computer Economics, "Economic Impact of Malicious Code Attacks," http://www.computereconomics.com/cei/press/pr92101.html, 2001.

[35] LBL-CONN-7, "Thirty Days' Wide-Area TCP Connections," http://ita.ee.lbl.gov/html/contrib/LBL-CONN-7.html, 2007.
[36] H.W. Hethcote, "The Mathematics of Infectious Diseases," *SIAM Rev.*, vol. 42, no. 4, pp. 599-653, 2000.
[37] M.A. Rajab, F. Monrose, and A. Terzis, "On the Effectiveness of Distributed Worm Monitoring," *Proc. Usenix Security Symp.*, pp. 225-237, 2005.
[38] M.A. Rajab, F. Monrose, and A. Terzis, "On the Impact of Dynamic Addressing on Malware Propagation," *Proc. ACM Workshop Rapid Malcode,* pp. 51-56, 2006.
[39] K. Rohloff and T. Basar, "Stochastic Behavior of Random Constant Scanning Worms," *Proc. IEEE Int'l Conf. Computer Comm. and Networks,* pp. 339-344, 2005.
[40] K. Rohloff and T. Basar, "The Detection of RCS Worm Epidemics," *Proc. ACM Workshop Rapid Malcode,* pp. 81-86, 2005.
[41] S.E. Schechter, J. Jung, and A.W. Berger, "Fast Detection of Scanning Worm Infection," *Proc. Int'l Symp. Recent Advances in Intrusion Detection,* pp. 59-81, 2004.
[42] S. Sellke, N. Shroff, and S. Bagchi, "Modeling and Automated Containment of Worms," *Proc. IEEE Int'l Conf. Dependable Systems and Networks,* pp. 528-537, 2005.
[43] E. Skoudis, *Counter Hack: A Step-by-Step Guide to Computer Attacks and Effective Defenses.* Prentice Hall, 2002.
[44] N. Weaver, S. Staniford, and R. Cunningham, "A Taxonomy of Computer Worms," *Proc. ACM Workshop Rapid Malcode,* pp. 11-18, 2003.
[45] N. Weaver, S. Staniford, and V. Paxson, "Very Fast Containment of Scanning Worms," *Proc. Usenix Security Symp.*, pp. 29-44, 2004.

Sarah H. Sellke received a degree in applied mathematics from Tsinghua University, Beijing, China, the MS degree in mathematics and the MS degree in electrical and computer engineering from Purdue University. She is a PhD candidate in the School of Electrical and Computer Engineering, Purdue University. She is an experienced software engineer. She was with Motorola Inc. and Canteliver Techonologies. She is the recipient of a US National Science Foundation graduate research fellowship and Purdue Ross graduate fellowship. Her research interests are in network security, information theory, and applied probability theory. She is a student member of the IEEE.

Ness B. Shroff received the PhD degree from Columbia University, New York, in 1994. He joined Purdue University immediately thereafter as an assistant professor, where he became a professor of the School of Electrical and Computer Engineering in 2003 and director of CWSA in 2004, a university-wide center on wireless systems and applications. He joined the Ohio State University in 2007 as the Ohio Eminent Scholar of Networking and Communications and professor of ECE and CSE. His research interests include wireless and wireline communication networks. He is especially interested in fundamental problems in the design, performance, control, and security of these networks. He is an editor for *IEEE/ACM Transactions on Networking* and the *Computer Networks Journal* and was an editor of *IEEE Communications Letters*. He has served on the technical and executive committees of several major conferences and workshops. He was the technical program cochair of IEEE INFOCOM '03, the premier conference in communication networking. He was also the conference chair of the 14th Annual IEEE Computer Communications Workshop (CCW '99), the program cochair for the symposium on high-speed networks, Globecom '01 and the panel cochair for ACM Mobicom '02. He was also a co-organizer of the NSF Workshop on Fundamental Research in Networking, held in Arlie House Virginia, in 2003. He will serve as the technical program cochair of ACM Mobihoc '08. He is a fellow of the IEEE. He received the IEEE INFOCOM '06 Best Paper Award, the IEEE IWQoS '06 Best Student Paper Award, the 2005 Best Paper of the Year Award for the *Journal of Commnications and Networking*, the 2003 Best Paper of the Year Award for *Computer Networks*, and the US National Sceince Foundation CAREER Award in 1996 (his INFOCOM 2005 paper was also selected as one of two runner-up papers for the best paper award).

Saurabh Bagchi received the MS and PhD degrees from the University of Illinois, Urbana-Champaign, in 1998 and 2001, respectively. He is an assistant professor with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, Indiana. He is a faculty fellow with the Cyber Center and has a courtesy appointment with the Department of Computer Science, Purdue University. He leads the Dependable Computing Systems Laboratory (DCSL), Purdue University, where he and a set of wildly enthusiastic students try to make and break distributed systems for the good of the world. His work is supported by the US National Science Foundation (NSF), Indiana 21st Century Research and Technology Fund, Avaya, Motorola, and Purdue Research Foundation, with equipment grants from Intel and Motorola. His papers have been runners up for the best paper in the 15th IEEE International Symposium on High Performance Distributed Computing (HPDC '06), 2005 International Conference on Dependable Systems and Networks (DSN), and MTTS 2005. He has been a member of the organizing committee and the program committee for DSN and the Symposium on Reliable Distributed Systems (SRDS). He is a senior member of the IEEE and the IEEE Computer Society.