This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2008 proceedings.

Exact Modeling of Propagation for Permutation-Scanning Worms

Parbati Kumar Manna Shigang Chen Sanjay Ranka
Department of Computer & Information Science & Engineering, University of Florida
{pkmanna, sgchen, ranka}@cise.ufl.edu

Abstract—**Modeling worm propagation has been an important research subject in the Internet-worm research community. An accurate analytical propagation model allows us to study the spreading speed and traffic pattern of a worm under an arbitrary set of worm/network parameters, which is often computationally too intensive for simulations. More importantly, it gives us an insight into the impact of each worm/network parameter on the propagation of the worm and the effectiveness of a potential defense mechanism that is designed to control some of those parameters. Traditionally, most modeling work in the area concentrates on the relatively simple random-scanning worms. However, worm technologies have advanced rapidly in recent years. By enabling close coordination among all infected hosts, the permutation-scanning worms minimize the duplication of effort when scanning the whole Internet address space. They propagate much faster, and more importantly, can be much more stealthy than the random-scanning worms. Modeling these worms, however, remains a challenge to date. This paper proposes a mathematical model that *precisely* characterizes the propagation patterns of the permutation-scanning worms. The analytical framework captures the interactions among all infected hosts by a series of inter-dependent differential equations, which together present the overall behavior of the worm. We use simulations to verify the numerical results from the model, and demonstrate how the model can be used to study the impact of various worm/network parameters on the propagation.**

I. Introduction

Computer worms interest the security analysts immensely due to their ability to infect millions of computers in a very short period of time [1]. In recent years, both sophistication and damage potential of worms have increased tremendously. In order to counter the threat [2], [3], [4], we need to look into both their content (for signatures) and propagation pattern (for Internet-scale behavior). The propagation characteristics of a worm shows what kind of network traffic would be generated by that worm and how fast must the response time be to counter it. Therefore, in order to understand (and possibly counter) the damage potential of worms, it is very important to characterize their overall propagation properties.

Although modeling worm propagation has been an active research area [5], [6], [7], [8], [9], one might question the practical importance of such work if it is possible to obtain fairly good approximation of the worm's propagation characteristics by running a simulator for a sufficient number of times and taking the average. However, there are reasons why simulations may not always be able to produce the intended results. First, it often takes a long time to simulate a single run of worm propagation for one set of worm/network parameters (16 hours in our case on a Intel Xeon 2.80GHz processor for 400M hosts that are estimated to be in today's IPv4 space). To learn the average behavior, many such runs need to be performed, and the whole simulation process has to be redone for any parameter change, *e.g.* for a different population size of vulnerable hosts or a different scanning speed of infected hosts. Second, the simulation overhead can be prohibitively high in some cases. Suppose we want to simulate a worm that exploits a commonly used Windows service on today's Internet. It means that the vulnerable population size could be in the order of several hundred millions as Windows machines dominate the Internet. If there are 300M such computers, they will entail 300M records in the simulation, one for each vulnerable host. Even if each record is one integer (keeping its address alone), it will require a memory of 1.2 GB. Now, if we want to study the effect of migration from IPv4 to IPv6 on worms, a full-scale simulation of scanning the address space of size 2^{128} will be computationally infeasible for a modest PC. In comparison, numerical computation based on a mathematical model takes little time to produce the detailed propagation curves. Third, simulation results themselves do not always give the *mathematical insight* that a formal model does. One may guess upon the impact of various parameters on worm propagation based on extensive simulations (which may take enormous time), but such guesses can never be as accurate and comprehensive as an analytical model, which tells exactly why and by how much a parameter change will affect the outcome.

Traditionally, most modeling work [7], [8] concentrates on the relatively simple random-scanning worms, which scan the Internet either randomly or with bias towards local addresses in order to reach all the vulnerable hosts. This strategy leaves a large footprint on the Internet (which reveals the worm's presence), and different infected hosts may end up scanning the same address repeatedly. In recent years, worm technologies have advanced rapidly to address these problems. By enabling close coordination among all infected hosts, the permutation-scanning worms (introduced in the seminal paper [8] by Staniford *et al.*) minimize the duplication of effort when

We would like to thank the anonymous reviewers for their suggestions that helped improve the paper. This work is supported in part by the National Science Foundation under Grant ITR 0325459 and NSF Grant number 0312038 (under a subcontract from FIU) and Cisco University Research Award (to Shigang Chen in 2007). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or Cisco.

scanning the Internet through a divide-and-conquer approach. There, each active infected host is responsible for scanning a subset of all addresses, and this subset may vary over time. Such a cooperation strategy empowers the worm with the ability to propagate either much faster, or alternatively, much stealthier (if the infected hosts scan at lower rates). Warhol worms, which are similar to permutation-scanning worms with larger hitlists, have been shown to be able to infect the whole of the Internet in a matter of minutes [8]. However, modeling these potent worms has remained a challenge to date.

In this paper, we propose a mathematical model that *precisely* characterizes the propagation patterns of the permutation-scanning worms. The analytical framework captures the interactions among all the infected hosts by a series of inter-dependent differential equations, which together present the overall behavior of the worm. We use simulations to verify the numerical results from the model, and show how the model can be used to assess the impact of various worm/network parameters on the propagation.

The rest of this paper is organized as follows. Section II describes the permutation-scanning worms. Section III introduces several important concepts underlying our mathematical model. Sections IV and V present the exact propagation models for the basic permutation-scanning worm and its general extension, respectively. Section VI shows the effects of different worm/network parameters on the worm propagation. Section VII draws the conclusion.

II. Anatomy of a Permutation-Scanning Worm

In this section, we explain how the permutation-scanning worms work. We first describe the divide-and-conquer nature of the permutation-scanning worms. We then discuss the reason for address permutation and the stealth potential of such worms, and conclude with the use of hitlists.

A. Divide-and-Conquer

To reduce the duplication of effort, the infected hosts may collaborate in dividing the IPv4 address ring into disjoint sections, each of which will be scanned by one host. Each initially infected host begins from its own location on the address ring and *sequentially* scans the addresses clockwise along the ring. Whenever it infects a host, it continues scanning the addresses after that host, while the newly infected host chooses a random location on the ring and starts to sequentially scan addresses clockwise after *that* location. When an active host h_1 hits an already infected host h_2, it knows that addresses after h_2 must have been scanned earlier by another active host that infected h_2, or by h_2 itself in case h_2 was one of the originally infected hosts to start with. In either case, h_1 jumps to a randomly location on the ring and starts to scan addresses clockwise after that location. An active host retires (stops scanning) after hitting a certain number of already-infected hosts.

An alternative to the above random-jump approach is to assign each infected host a section of the address ring for scanning. As a host sequentially scans its section, when it infects another host, it assigns half of its remaining unscanned address section to the latter and adjusts its own section boundary accordingly. When a host reaches the end of its section, it retires. The problem with this approach is that it is not fault-tolerant. If one infected host is blocked out or somehow crashes, its remaining section will not be scanned. Random jumps (as mentioned above) help solving this problem. This paper will focus on random-jump worms only.

B. Permutation

While the above divide-and-conquer method maintains a much smaller network footprint by minimizing duplication of scanning, it has a serious weakness. Since the IP addresses scanned by an infected host are contiguous, it is susceptible to be identified by address-scan detectors or other IDSs that look for worms performing local subnet scanning. To counter this, Staniford *et al* [8] showed that a worm can permute the IP address space into a virtual one (called the *permutation ring*) through encryption with a key. The divide-and-conquer method is then applied on this permutation ring. While each infected host still goes through contiguous addresses on the permutation ring, it actually scans the IP addresses that the permuted addresses are decrypted to, which cannot be easily picked up by address-scan detectors because those IP addresses are pseudo-random and distributed all over the Internet.

C. Stealth

Fast propagation and stealth are two conflicting goals that the worm designers strive to balance. To spread fast, infected hosts should scan at high rates, which however makes them easier to be detected [3], [4], [1]. To be stealthy, they have to act as normal as possible, scanning the Internet at a controlled low rate, which is a worm parameter that can be set before release. A stealthy worm can be more harmful. A fast worm generates headline news, such as Slammer [1] that caused widespread network congestion across Asia, Europe and Americas. Such a worm is more likely to be detected quickly and attract defense resources to react fast for its elimination. A stealthy worm propagates slower but may stay undetected for a long time, potentially doing more harm.

D. Hitlist

The initial part of worm propagation is most time-consuming, as only a few infected hosts perform scanning in a vast address space. Once the number of infected reaches a critical mass, the rate of new infections goes up drastically. To improve the initial scanning speed of a stealthy worm, one can use a *hitlist* as proposed in [8], which is a pre-compiled list of target addresses that are very likely to be vulnerable, *e.g.*, a list of hosts with port 80 open for a worm targeting at a certain type of web servers. During the hitlist-infection phase, the very first infected host starts scanning the IP addresses in the hitlist, and whenever it can infect one, it gives away half of the remaining hitlist to the newly infected host so that together they can infect all the hosts in the original hitlist quicker. This process repeats, and as a result, if v out of the S addresses in the hitlist turn out to be actually vulnerable hosts, all those

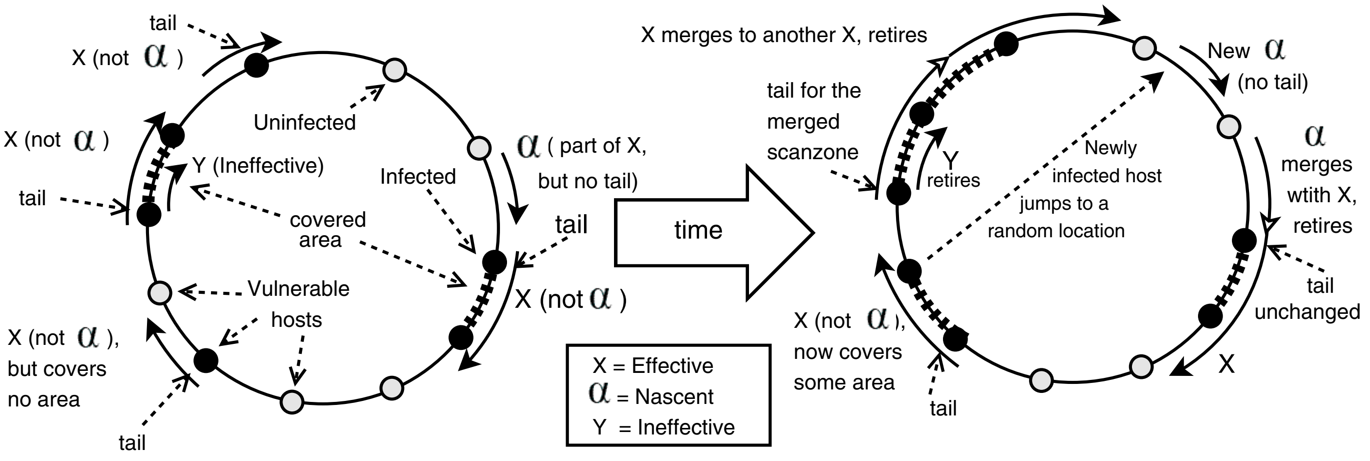

tail
X (not α)
X (not α)
tail
Uninfected
Y (Ineffective)
Infected
α (part of X, but no tail)
covered area
tail
Vulnerable hosts
X (not α)
X (not α), but covers no area
tail
time
X = Effective
α = Nascent
Y = Ineffective
X merges to another X, retires
New α (no tail)
tail for the merged scanzone
Y retires
Newly infected host jumps to a random location
α merges wtith X, retires
tail unchanged
X (not α), now covers some area
tail
X

Fig. 1. Depiction of scanzones for a 0-jump worm at two time points. Scanzones of active hosts are depicted as arcs on the permutation ring. Uninfected and infected vulnerable hosts are depicted as white and dark dots on the permutation ring, respectively.

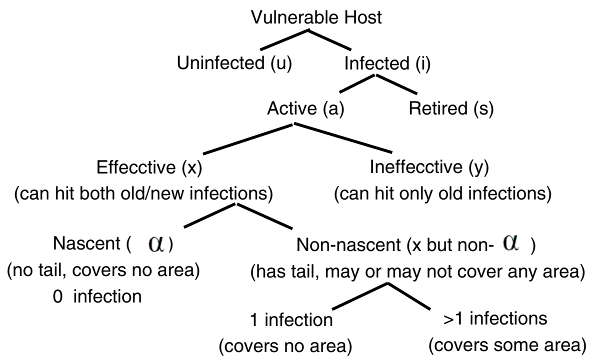


Fig. 2. The classification of the vulnerable hosts for a permutation-scanning worm

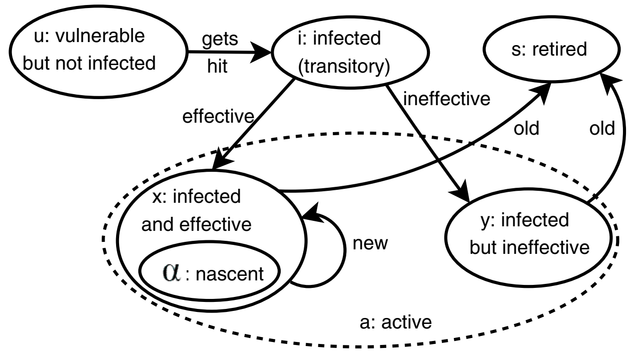


Fig. 3. State Diagram of a 0-jump worm. Here, "new" or "old" indicates the event of a *new* or *old* infection. Similarly, "ineffective" or "effective" indicates whether the newly spawned host, after the random jump, lands in an area that is already "covered" or not.

hosts will get infected in $O(\frac{S}{vr}\log_2 v)$ time, where r is the scanning rate. Even for a modestly big hitlist, this time is miniscule compared to the time it will take to infect the rest of the vulnerable hosts outside the hitlist. To illustrate with an example, suppose there are about 1M vulnerable hosts in IPv4 and a worm starts with a hitlist of S = 10K hosts, with approximately v = 5K of them actually being vulnerable. If the scanning rate r is 1000 scans/sec, then the time taken to infect the initial 5K hosts in the hitlist would be approximately 0.025 second, which can arguably be ignored compared to the time the worm will take to infect the rest of the vulnerable hosts in the Internet. Thus, to keep the model simple, if the hitlist contains v vulnerable hosts, we assume that all v of them are infected at the beginning (time t=0).

III. Scanzone and Scanning Efficiency

In this section, we introduce the concept of *scanzone*, and then show how we can analyze an infected host's efficiency (ability to potentially generate new infection) from its scanzone. We conclude with a formal classification of the infected hosts based on their efficiency.

A. Terminology

We begin by defining the basic terminology used in this paper. We classify infected hosts into two categories: (1) *active infected hosts*, which are actively scanning for vulnerable hosts, and (2) *retired infected hosts*, which have stopped scanning. When the context makes it clear, we omit "infected" from the above terms. The rest of the terms are defined as follows:

Jump: When an infected host chooses a random location on the permutation ring to begin its sequential scan along the ring, we say that the host *jumps*.

Old Infection: When an active host hits a vulnerable host h that was infected previously, we denote the event (as well as host h) as an *old infection*.

New Infection: When an active host hits a vulnerable host h that was *not* previously infected, we denote the event (as well as host h) as a *new infection*.

***k*-Jump Worm**: A permutation-scanning worm is called a k-jump worm if an active host, upon hitting an old infection, jumps to a new location on the permutation ring to resume scanning, but it will retire when hitting its $(k{+}1)^{th}$ old infection. When a vulnerable host not in the hitlist becomes a new infection, it jumps to a random location on the ring to begin its scan. Subsequently this host can make k other jumps after hitting old infections on the ring. For a vulnerable host in the hitlist, it begins scanning from its own location and then it can make k jumps.

0-Jump Worm: A permutation-scanning worm is called a 0-jump worm if an active host retires upon hitting its very first old infection. It is a special case of k-jump worm with $k{=}0$. A vulnerable host not in the hitlist can make one jump when it becomes a new infection itself, but subsequently when it hits an old infection, it will retire immediately.

B. Scanzone of an Active Infected Host

As an active infected host h scans the addresses along the permutation ring, it leaves behind a contiguous section of scanned addresses. This contiguous section, called the *scanzone* of host h, contains the addresses that h has scanned since its last jump or time 0 if h has not jumped yet; it may contain more addresses if scanzone merge happens, which will be discussed shortly. Together the scanzones of all active hosts cover all addresses scanned so far. The address of each infected host belongs to a scanzone because it is a scanned address. The *front end* of a scanzone is the address that is currently being scanned by h; the *back end* refers to the address at the other end of the scanzone. Evidently all vulnerable hosts in a scanzone must have been infected. Among all infected hosts in a scanzone, the one that is closest to the back end is called the ***tail*** of the scanzone, and the one that is closest to the front end is called the *head* of the scanzone. The portion of a scanzone between the tail and the head is referred to as the ***covered area*** (portrayed as in Figure 1) of the scanzone. A scanzone may not have a tail (or head) if the active infected host has not hit any vulnerable host since its last jump, and it will have zero covered area if it does not have at least two infected hosts in it.

As h scans more and more addresses, the front end advances to expand the scanzone. But when h hits an old infection h_{old} (which must belong to the scanzone of some active infected host h_1), h surrenders its scanzone by merging it to h_1's scanzone. Then h jumps to a random location to create its new scanzone afresh, or retires if h_{old} is the $(k{+}1)^{th}$ old infection that it hits. Therefore, the back end of a scanzone may also change if the front end of another scanzone catches up its tail and causes a merge. Merges create larger scanzones. Eventually, all scanzones will be merged into one when all active hosts retire. We recall that only active hosts have scanzones (uninfected or retired hosts do not). We must stress that an infected host does not need to know its scanzone; it is an abstract concept used in our mathematical modeling only. The scanzones are shown as arcs on the permutation ring in Figure 1, which also illustrates other concepts to be defined in this section.

C. Classification of Vulnerable Hosts

In our model, we define classes u, i, a, s, x, y, α for vulnerable hosts that are uninfected, infected, active, retired, effective, ineffective, and nascent, respectively, and we deliberately make the above class notations the same as the corresponding variables in our later propagation model for the *sizes* of these classes.

We classify the active hosts into subcategories by judging each active host's *effectiveness of scanning*, which is the ability of generating new infections *before* hitting an old one (note that every active host will eventually hit an old infection). The classification of active infected hosts is given below (Figure 2 showing the complete classification tree):

- **Ineffective (class y)**: An active infected host is considered *ineffective* if it is impossible for the host to generate any new infection in future before hitting an old one. An active host that jumps into a covered area to begin its scanning is evidently ineffective since its first hit will always be an old infection.
- **Effective (class x)**: An active infected host is considered effective if it can *potentially* generate a new infection in future *before* it hits an old one. When an infected host jumps to a point outside of all covered areas and starts scanning from that point on, it can potentially generate new infections. Thus, it is called *effective*, and is branded as class x. This class is further subdivided as follows:
  - **Nascent (class α):** Those effective hosts that are yet to infect any vulnerable host in their *current* scanzone (thus have no tail) are termed as *nascent* (class α). An active host becomes nascent after it takes a jump and lands outside covered area, since after the jump it starts with a fresh scanzone.
  - **Non-Nascent Effective (non-α class x):** Once a nascent host hits a new infection, it becomes a non-nascent effective host; and the host it just infected becomes the tail of its scanzone. Also, each of the initially infected hosts starts as a non-nascent effective host because its scanzone has a tail from the very beginning (the active host itself).

We observe that every infected host in the address space belongs to the scanzone of a non-nascent effective host. This statement is true at the beginning as each of the initially infected hosts belongs to its own scanzone. Later, when a non-nascent effective host h_1 infects some host h_{new}, h_{new} becomes part of h_1's scanzone. When h_1 retires by hitting h_{old} (tail of another non-nascent effective host h_2's scanzone), h_1's scanzone merges with h_2's scanzone and the infections in h_1's scanzone now become part of h_2's scanzone. Continuing this way, every infected host remains part of the scanzone of a non-nascent effective host until the last active host retires. It must be noted that we did not need to consider the retirement or transitions of nascent or ineffective hosts since their scanzones do not contain any infected hosts.

Figure 3 gives the class transition diagram for a 0-jump worm. A vulnerable host becomes infected when it is scanned by another infected host. When it jumps, it may be either effective or ineffective (if it jumps to a covered area). An effective host begins as a nascent one and becomes non-nascent once it infects another host. An active host retires upon hitting an old infection. Figure 1 also provides illustration for transitions among different classes.

In the following two sections, we will first model the 0-jump worms and then model the general k-jump worms.

IV. Modeling the Propagation of 0-Jump Worms

In this section, we derive a series of differential equations that together form the propagation model of 0-jump worms. We extend it for k-jump worms in the next section.

A. Important Quantities in Modeling

The propagation model of a worm reflects the fractions of vulnerable hosts that are infected, active and retired over time. A scan message that does not hit any vulnerable host does not change these numbers. Thus, it is evident that the modeling needs to be based on the event of a scan message hitting a vulnerable host only. When that event happens, all the aforesaid numbers change; we derive the model by analyzing the precise amounts by which they change. To model a 0-jump worm mathematically, we must be able to compute the following quantities:

Q1: Between time t and $t+dt$ (for an infinitesimally small dt), how many vulnerable hosts is an active host expected to hit by its scan messages?

Q2: When an effective host hits a vulnerable host h, what is the probability that h is an old infection, and what is the probability that h is a new infection? Note that an ineffective host never hits a new infection.

Q3: After a newly infected host jumps, what is the probability for it to be ineffective and what is the probability for it to be effective?

B. Determing the Quantities Using Probabilistic Approach

Let N be the size of the address space, V the total number of the vulnerable hosts, r the scanning rate and v the number of the vulnerable hosts in the hitlist of a permutation worm.

We use $u(t)$, $i(t)$, $a(t)$, $s(t)$, $x(t)$, $y(t)$ and $\alpha(t)$ to denote the fractions of vulnerable host population that are uninfected, infected, active, retired, effective, ineffective and nascent at time t, respectively. From Figure 2, it is easy to see that $u(t)+i(t)=1$, $i(t)=a(t)+s(t)$, and $a(t)=x(t)+y(t)$.

Answer for Q1: Let $\boldsymbol{f_{hit}}$ be the number of vulnerable hosts that an active host is expected to hit during a period of dt after time t. Since vulnerable hosts are uniformly distributed in the permuted address space due to randomization of the permutation process, every address on the permutation ring has a probability of $\frac{V}{N}$ to be a vulnerable host. An active host scans $r \times dt$ addresses during dt period. Hence, we have $f_{hit} = r \times dt \times \frac{V}{N}$. Note that the vulnerable hosts that are hit may include both new and old infections.

Answer for Q2: When an effective host hits a vulnerable host, let $\boldsymbol{f_{new}(t)}$ ($\boldsymbol{f_{old}(t)}$) denote the probability for the vulnerable host to be a new (old) infection. We observe that an effective host can hit only two types of vulnerable hosts: 1) those that are uninfected, and 2) infected ones that are the *tails* of scanzones for non-α effective hosts. Recall that scanzones of nascent or ineffective hosts do not have tails. At time t, there are $V(1-i(t))$ uninfected vulnerable hosts (possible new infections) and $V(x(t)-\alpha(t))$ tails (possible old infections). Hence, the chance for hitting a new infection is $f_{new}(t) = \frac{V(1-i(t))}{V(1-i(t))+V(x(t)-\alpha(t))} = \frac{(1-i(t))}{(1-i(t))+(x(t)-\alpha(t))}$, and $f_{old}(t) = 1 - f_{new} = \frac{(x(t)-\alpha(t))}{(1-i(t))+(x(t)-\alpha(t))}$.

Answer for Q3: After a newly infected host jumps to a random location to begin its scanning, let $\boldsymbol{f_{ineff}(t)}$ ($\boldsymbol{f_{eff}(t)}$) be the probability for the host to be ineffective (effective). As a host becomes ineffective when it jumps into a covered area, $f_{ineff}(t)$ must be equal to the fraction of the permutation ring that all covered areas together represent. Because vulnerable hosts are distributed randomly on the ring, it must also be equal to the fraction of vulnerable hosts that are located in the covered areas, excluding tails because, if we use the number of vulnerable hosts in a covered area to represent its length (in a statistical sense), we cannot count both head and tail that delimits the two ends of the area. All infected hosts, $Vi(t)$ of them, are located in the covered areas, and there are $V(x(t)-\alpha(t))$ tails (single-infection scanzones can be thought of each having a covered area of length 0) Therefore, $f_{ineff}(t) = \frac{Vi(t)-V(x(t)-\alpha(t))}{V}$, and $f_{eff}(t) = 1 - f_{ineff}(t)$.

C. Propagation Model

We now derive how $i(t)$, $a(t)$, $s(t)$, $x(t)$, $y(t)$ and $\alpha(t)$ change over time t. Below we compute the amounts, $di(t)$, $da(t)$, $ds(t)$, $dx(t)$, $dy(t)$ and $d\alpha(t)$, by which they change respectively over an infinitesimally small dt after time t. This will give us a set of differential equations that together characterize the propagation of 0-jump worms.

- $\boldsymbol{di(t)}$: It is the number of new infections over dt. Only effective (class x) hosts can hit new infections. The number of vulnerable hosts hit by effective hosts over dt is $x(t)\ f_{hit}$, and each of them has a probability of $f_{new}(t)$ to be a new infection. Hence $di(t) = x(t)\ f_{hit}\ f_{new}(t)$.
- $\boldsymbol{dx(t)}$: Each of the $x(t)f_{hit}f_{new}(t)V$ new infections has a probability of $f_{eff}(t)$ to be effective. This adds $x(t)f_{hit}f_{new}(t)Vf_{eff}(t)$ new effective hosts after dt. On the other hand, effective hosts hit $x(t)\ f_{hit}\ f_{old}(t)V$ old infections during dt, each causing an effective host (that hits the old infection) to retire. Combining the above two numbers and representing the gross change in fraction, we have $dx(t) = x(t)\ f_{hit}\ f_{new}(t)\ f_{eff}(t) - x(t)\ f_{hit}\ f_{old}(t)$.
- $\boldsymbol{d\alpha(t)}$: Each nascent host (which is effective by defintion) is no longer nascent once it hits any vulnerable host. Each of its $r \times dt$ scan messages has a $\frac{V}{N}$ probability of hitting a vulnerable host. Hence, the probability for a nascent host to become non-nascent over dt is $r \times dt \times \frac{V}{N} = f_{hit}$ because, as dt approaches to zero, the

joint probabilities for two or more hits is negligible. This reduces the number of nascent hosts by $\alpha(t)Vf_{hit}$. On the other hand, since all new effective hosts created during dt start as nascent, we have $x(t)V\ f_{hit}\ f_{new}(t)\ f_{eff}(t)$ new nascent hosts. Combining these two numbers and representing the gross change in fraction, we have $d\alpha(t) = x(t)\ f_{hit}\ f_{new}(t)\ f_{eff}(t) - \alpha(t)\ f_{hit}$.

- $\boldsymbol{dy(t)}$: Recall that whenever a host jumps into a covered area, it becomes ineffective. For a 0-jump worm, only the newly infected hosts make a jump and thus only they may increase $y(t)$. There are $x(t)Vf_{hit}f_{new}(t)$ new infections, and each has a probability of $f_{ineff}(t)$ to become ineffective. On the other hand, when an existing ineffective host hits a vulnerable host, it retires since ineffective hosts can hit old infections only. Combining these two factors and representing the gross change in fraction, we have $dy(t) = x(t)f_{hit}f_{new}(t)f_{ineff}(t) - y(t)f_{hit}$.
- $\boldsymbol{ds(t)}$: Whenever an effective host hits an old infection, or an ineffective host hits *any* vulnerable host (which must be an old infection), it retires. Within time dt, there are $x(t)Vf_{hit}f_{old}(t) + y(t)Vf_{hit}$ newly retired hosts, and thus $ds(t) = x(t)f_{hit}f_{old}(t) + y(t)f_{hit}$.

Combining, we get the following equations:

$$\begin{aligned}
f_{hit} &= r \times dt \times \frac{V}{N} \\
f_{old}(t) &= \frac{x(t) - \alpha(t)}{1 - i(t) + x(t) - \alpha(t)} \\
f_{new}(t) &= \frac{1 - i(t)}{1 - i(t) + x(t) - \alpha(t)} = 1 - f_{old}(t) \\
f_{ineff}(t) &= i(t) - (x(t) - \alpha(t)) \\
f_{eff}(t) &= 1 - i(t) + x(t) - \alpha(t) = 1 - f_{ineff}(t) \\
di(t) &= x(t)\ f_{hit}\ f_{new}(t) \\
dx(t) &= x(t)\ f_{hit}\ f_{new}(t)\ f_{eff}(t) - x(t)\ f_{hit}\ f_{old}(t) \\
d\alpha(t) &= x(t)\ f_{hit}\ f_{new}(t)\ f_{eff}(t) - \alpha(t)\ f_{hit} \\
dy(t) &= x(t)\ f_{hit}\ f_{new}(t)\ f_{ineff}(t) - y(t)\ f_{hit} \\
ds(t) &= x(t)\ f_{hit}\ f_{old}(t) + y(t)f_{hit} \\
da(t) &= dx(t) + dy(t)
\end{aligned}$$

Finally, we add the incremental figures like $i(t+dt) = i(t) + di(t)$, $x(t+dt) = x(t) + dx(t)$ etc. The boundary conditions for the set of equations above are: $i(0) = a(0) = x(0) = \phi = \frac{v}{V}$, and $\alpha(0) = s(0) = y(0) = 0$, where ϕ is the number of vulnerable hosts in the hitlist (v) as a fraction of V.

D. Verification of Our Model

To generate the propagation graphs (G_1) from our model, we solve the differential equations numerically for $i(t)$ (infected), $a(t)$(active) and $s(t)$(retired) for increasing values of t using the boundary conditions defined earlier. To verify this result, this output needs to be matched with the actual propagation curves (G_2) of a permutation-scanning worm. Due to unavailability of a full-scale real-life propagation data, we code a worm simulator that mimics the behavior of a permutation-scanning worm on a permutation ring to generate G_2. Our simulator implements full network connectivity with all scan packets taking equal time to reach their target host. We run this simulator on an Intel P4 2.4 GHz processor over one thousand rounds and take the average. For every round, a different seed is used to initialize the pseudo-random number generator responsible for calculating the target address while taking a random jump. We verify our model by comparing G_1 and G_2 for different sets the worm/network parameters (N, V, v and r) and observe near-complete overlaps in *all* cases. The comparison results for one such set is shown in Figure 4.

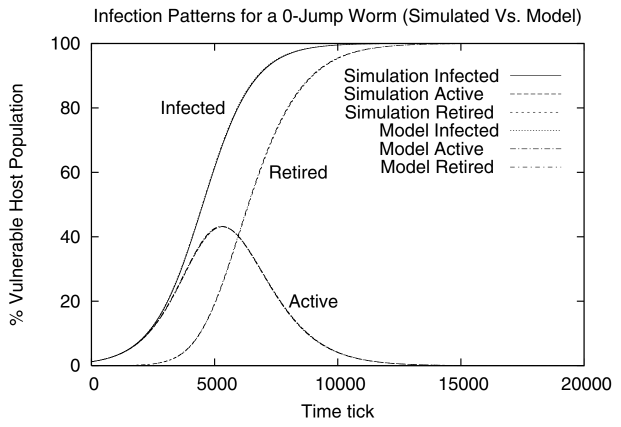


Fig. 4. Juxtaposition of propagation patterns of a 0-jump worm in simulation vs. according to the analytical model. We use $N = 2^{23}$, $V = 2^{13}$, $v = 100$ and scan rate $r = 1$ scan per time tick. The curves from the model and the curves from the simulation appear to be nearly indistinguishable.

V. Extending the Model to k-jump Worms

In this section, we demonstrate the flexibility of our analytical model by extending it to the k-jump worm. Modeling the propagation for a k-jump worm is important as it leads to a better understanding of the Warhol worm, which can infect the whole of Internet in a matter of minutes [8]. Warhol worms are similar to a permutation-scanning k-jump worm with a big hitlist and possibly with a larger value of k.

A. Difference Between 0-Jump Worm and k-Jump Worm

We begin with noting a subtle distinction in the nomenclature for k-jump worm compared to its 0-jump predecessor. In the 0-jump model, at time t none of the $a(t)$ active hosts have hit any old infection. However, for a k-jump worm, at time t any active host (class x, α and y) could have hit anywhere between 0 to k old infections. Therefore, while the terms $x(t)$, $\alpha(t)$ and $y(t)$ continue to denote the *total* fraction of vulnerable hosts that are effective (class x), nascent (class α) and ineffective (class y) at time t for a k-jump worm, each of those classes is further subdivided into $k+1$ subclasses depending on how many old infections they have already hit (between 0 and k). For example, class x is subdivided into classes x_0, x_1, x_2 … x_{k-1}, x_k such that $x(t) = \sum_{j=0}^{k} x_j(t)$, and similar notations are used for class α and y. For the ease of reference, the active hosts having already hit j old infections

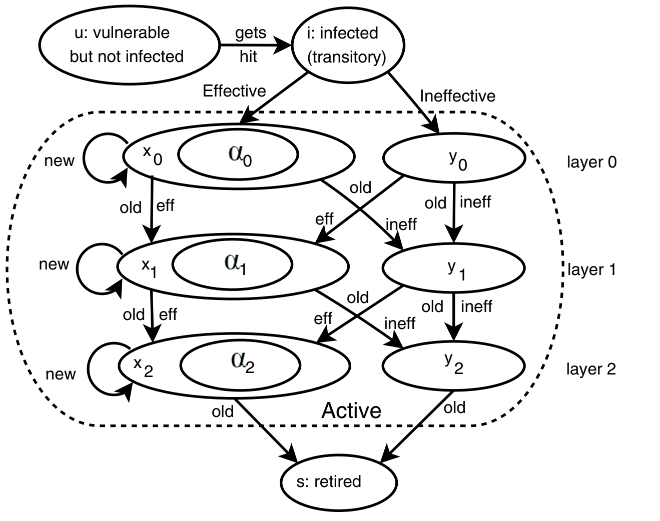


Fig. 5. State Diagram of a k-jump worm with k=2. The layer number indicates the number of old infections hit by that host till that time. Once the host hits its k+1th (in this case 3rd) old infection, it retires immediately.

are referred to as j-layer hosts. For example, the total number of nascent hosts that have hit 2 old infections till time t are denoted by $\alpha_2(t)$. We observe that for calculating the probabilistic figures $f_{old}(t)$, $f_{new}(t)$, $f_{eff}(t)$ and $f_{ineff}(t)$, this subdivision is immaterial since the only thing that matters for their calculation is how many infected, effective and nascent hosts are there in total at time t. So, the equations for deriving those figures remain unchanged.

B. Interaction among Scanning Hosts at Different Layers

The state diagram of the k-jump worm (for k=2) is depicted in Figure 5. The transitions between different classes in different layers are explained by the following observations:

- An active infected host never changes its layer by hitting a *new* infection. This is because the layer of a host indicates how many *old* infections the active host has hit till that time, and hitting a new infection does not change that. However, when it hits an old infection, it takes a jump, moves to the next layer and becomes either nascent or ineffective depending on whether it jumps into a *covered area* or not. However, if it was already at the k-layer, then it retires after hitting its $(k+1)^{th}$ old infection.
- Active hosts from *any* layer can hit a new infection. Therefore, for calculating change in $x_0(t)$, $\alpha_0(t)$ and $y_0(t)$, we must consider the new infections caused by effective worms from all the k+1 layers.
- For any layer other than the 0-layer, the incremental changes are caused by active hosts from the previous and the current layer only. The number of hosts in a layer increases when hosts in the previous layer hit old infections and move up to the current layer. Similarly, it decreases when hosts in current layer hit old infections and transition into the next layer. Therefore, all the computations for j-layer hosts (where $j \geq 1$) involve figures from layer j and layer $j-1$ only.

C. The Final Model of Propagation

Here we lay down the equations that model the propagation pattern for the k–jump worm. For the purpose of brevity, all the symbols used are function of time t; except f_{hit}, V and N, which are independent of time. For example, f_{new} denotes $f_{new}(t)$, $d\alpha_j$ denotes $d\alpha_j(t)$ and so on. We do not rewrite the equations for $f_{old}(t)$, $f_{new}(t)$, $f_{eff}(t)$ and $f_{ineff}(t)$ since they are the same as in the model for 0-jump worm.
$\forall j = 0 \ldots k$, we have

$$dx_j = \begin{cases} \text{if } j = 0, & x f_{hit}\, f_{new}\, f_{eff} - x_j\, f_{hit}\, f_{old}\ ; \\ \text{if } j > 0, & x_{j-1} f_{hit}\, f_{old}\, f_{eff} - x_j\, f_{hit}\, f_{old} \\ & + y_{j-1}\, f_{hit}\, f_{eff}\ ; \end{cases}$$

$$d\alpha_j = \begin{cases} \text{if } j = 0, & x f_{hit}\, f_{new}\, f_{eff} - \alpha_j\, f_{hit}\ ; \\ \text{if } j > 0, & x_{j-1} f_{hit}\, f_{old}\, f_{eff} - \alpha_j\, f_{hit} \\ & + y_{j-1}\, f_{hit}\, f_{eff}\ ; \end{cases}$$

$$dy_j = \begin{cases} \text{if } j = 0, & x f_{hit}\, f_{new}\, f_{ineff} - y_j\, f_{hit}\ ; \\ \text{if } j > 0, & x_{j-1} f_{hit}\, f_{old}\, f_{ineff} - y_j\, f_{hit} \\ & + y_{j-1} f_{hit}\, f_{ineff}\ ; \end{cases}$$

Finally, we define the other incremental figures:
$dx = \sum_{j=0}^{k} dx_j(t)$; $dy = \sum_{j=0}^{k} dy_j(t)$; $d\alpha = \sum_{j=0}^{k} d\alpha_j(t)$; $di = x\ f_{hit}\ f_{new}$; $da = dx + dy$; $ds = x_k f_{hit}\ f_{old} + y_k f_{hit}$;

We do not mention the rest of the equations like $x_j(t{+}dt)$ = $x_j(t) + dx_j(t)$, $s(t + dt) = s(t) + ds(t)$ etc. to maintain conciseness. The boundary conditions at time $t = 0$ are:

$i(0) = a(0) = x(0) = x_0(0) = \phi = \frac{v}{V}$. All the other counts ($s$, $x_1 \ldots x_k$, α, $\alpha_0 \ldots \alpha_k$, y, $y_0 \ldots y_k$ etc.) are zero at t=0.

D. Verification of the Correctness of the Model

We compare the result of the numerical model with actual worm simulation for different values of k in Figure 6 using the same experimental setup as described in Section IV-D. In all the cases, the model and the simulation overlap to yield nearly identical propagation graphs.

VI. Usage of the Analytical Model

In this section, we first describe the benefits of having an analytical model compared to running a simulator. Then, we analyze our model to see what kind of effects does each worm/network parameter (network size, vulnerable population size etc.) have on the propagation curves.

A. Analytical Modeling or Simulation?

Proper simulation of the Internet is very difficult due to its scale, heterogeneity and dynamics [10]. Even for a rather simplified version of the Internet, without an analytical model one would need to take the average of multiple runs of a simulator in order to get acceptably reliable propagation curves. And since each run could potentially take a long time for big values of N and V, the whole process could take an enormous amount of time. In our experimental setup, it took 16 hours on a Intel Xeon 2.8 GHz processor with 4GB of RAM to run a *single* run of a simulation of 400M vulnerable hosts (as in Internet today) on IPv4 for one set of worm/network parameters. In order to run the same simulation for IPv6 ($N = 2^{128}$), it is easy to see that the runtime would be

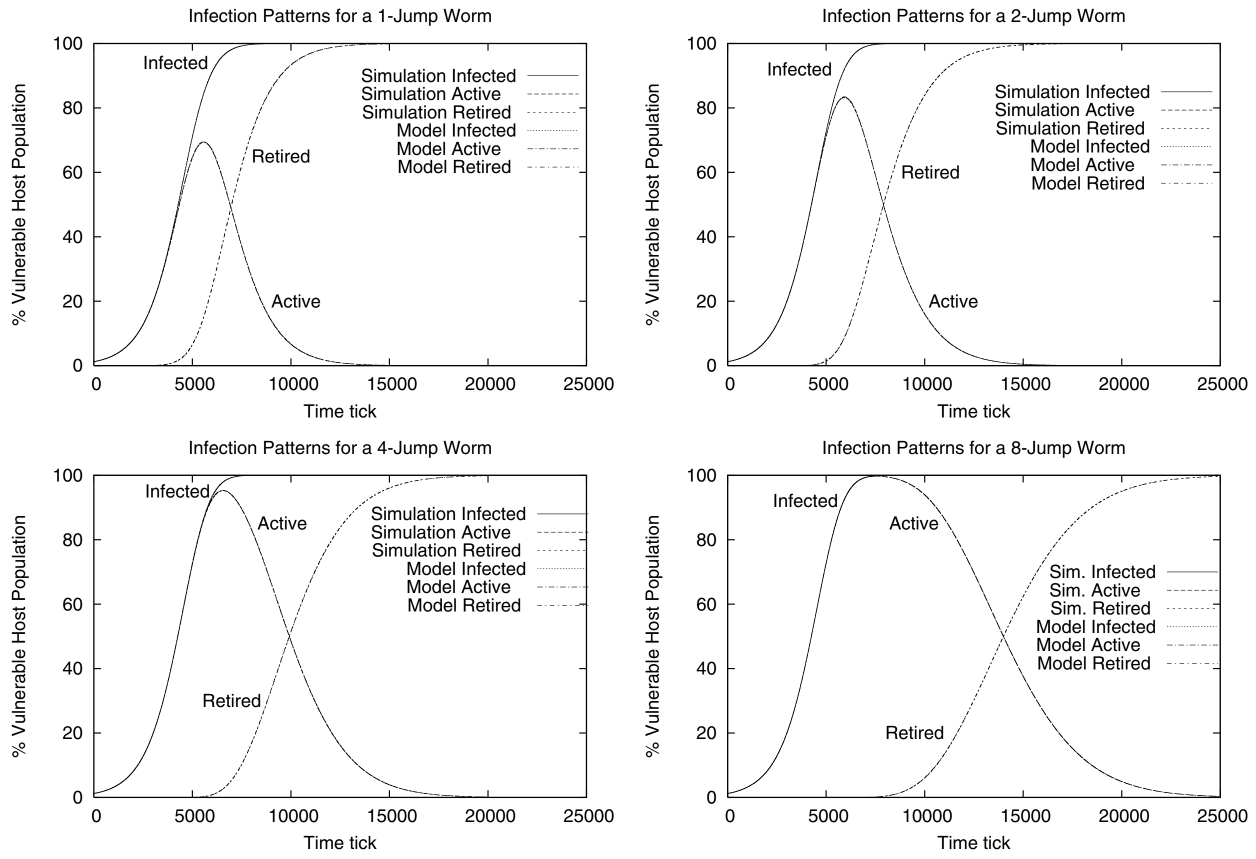


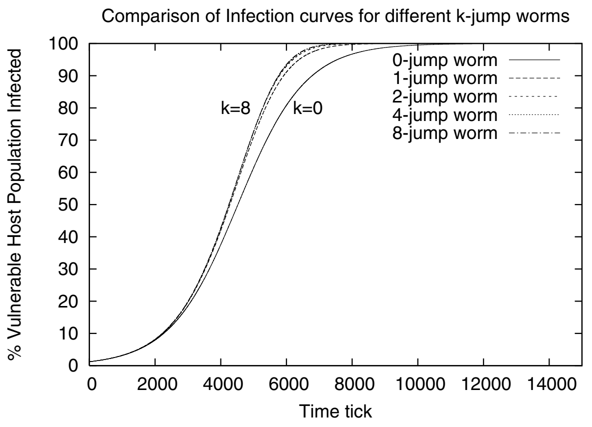
Fig. 6. Juxtaposition of the propagation patterns of different k-jump worms (for $k = 1, 2, 4$ and 8) obtained via simulation vs. obtained by the analytical model for address space size $N = 2^{23}$, vulnerable host population size $V = 2^{13}$, scanning rate $r = 1$ scan per time tick and the hitlist containing $v = 100$ vulnerable hosts. In all the cases, the propagation patterns overlap completely.

astronomical. On the other hand, a *single* run of the numerical simulation of the analytical model, which takes just seconds to run, gives us the correct results. Moreover, the effect of increasing the worm/network parameters (like N and V) on runtime is insignificant for a numerical solver compared to the effect it has on an actual worm simulator. While arguments can be made for doing a scaled-down simulation and then simply scaling up the results, such simulations are often not fully accurate and suffer from stochastic fluctuations and other problems [5]. Moreover, such simulations cannot predict with confidence what *precise* effect each worm/network parameter will have on the overall outcome, and for what reason. On the other hand, an analytical model can tell *exactly* why and by how much would a parameter affect the outcome, to the extent of predicting whether a scaled-down simulation would preserve the propagation characteristics or not.

B. Effects of Worm/Network Parameters on Propagation

Here we analyze the exact effect of each worm/network parameters on the propagation.

- **Effect of Address Space Size (N)**: The only term that is directly affected by N is $f_{hit} = r \times dt \times \frac{V}{N}$. Since all the incremental terms (like $dx(t)$) are direct multiples of f_{hit}, the growth rates of all the curves (infected, active and retired) are inversely proportional to N. Therefore, if the size of the network is increased p times while keeping all other parameters constant, time to reach every milestone in the original graph will also increase p-fold exactly. This is why transition to IPv6 is important.
- **Effect of Vulnerable Host Population Size (V)**: The only terms that are affected by V are $f_{hit} = r \times dt \times \frac{V}{N}$, and $\phi = \frac{v}{V}$ (in the boundary condition). Thus, a p-fold increase of V results in a p-fold reduction in propagation time, as long as the hitlist is also increased p-fold. If the hitlist size remains the same, then an increased V implies decreased ϕ, which means lower rate of infection initially. However, the increased probability of getting a hit ($\frac{V}{N}$) more than compensates the initial deficit. Thus, a bigger vulnerable population means faster infection.
- **Effect of Hitlist Size (v)**: The effect of changing v has already been discussed in conjunction with V. However, the effect of changing v for a fixed N and V is more important as it is completely under the control of the worm-author. As shown by our analytical model, a higher v causes the origin in the propagation graph to shift to the right, which implies faster infection. Moreover, a larger hitlist can shorten the initial slow-infection period significantly. In our experiments with a 0-jump worm using 1% of V as the hitlist, it takes the same amount of time to reach from 50% infection level to 67% as from 1% to 2%. Thus, one can achieve a considerable gain in



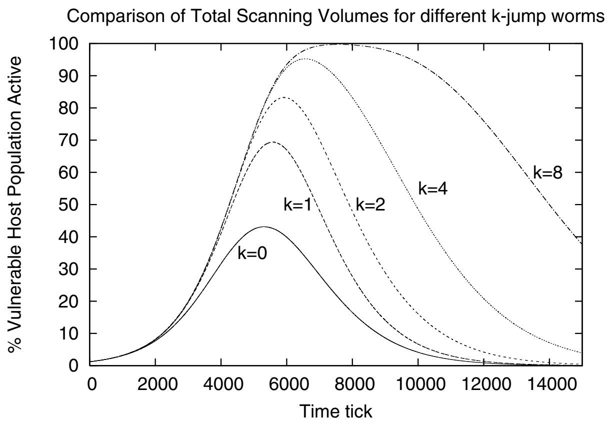


Fig. 7. Comparison of infection speed and total scanning volume for various k-jump worms for $N{=}2^{23}$, $V{=}2^{13}$, $v{=}100$, and scanning rate $r{=}1$ scan per time tick. The scanning volume is defined as the area under the active curve. The infection speed increases with k, but for increasingly higher value of k, the rate of increase diminishes. On the other hand, the scanning volume increases *significantly* with increasing k.

the infection time just by increasing the hitlist size.

- **Effect of Scanning Rate (r)**: The only term that is affected by r is $f_{hit} = r \times dt \times \frac{V}{N}$. Since all the incremental terms on the equation (like $dx(t)$, $d\alpha(t)$ etc.) are direct multiples of f_{hit}, the infection time is inversely proportional to the scanning rate. Thus, if the scanning rate is doubled, the infection time will be halved.
- **Effect of Varying k for a k-Jump Worm**: Figure 7 shows that increasing value of k helps to achieve faster infection. However, beyond a certain value of k (8 in this case), the incremental gain is negligible. On the other hand, with higher values of k, the onset of retirement for active hosts happens at increasingly later time. In fact, for $k{=}8$ in our experimental setup, almost *all* the infected hosts are active when we achieve nearly full infection, which implies a big network footprint. Therefore, it makes little sense to compromise stealth by deploying a k-worm with a very high value of k.

VII. Limitation, Extension and Conclusion

In this paper, we have successfully modeled the propagation characteristics of the permutation-scanning worm, a worm with a very high damage potential. In order to design this model, we have introduced the concept of scanzones, which is a completely novel way to understand the scanning dynamics of sequentially-scanning worms. By extending our model to different varieties of permutation-scanning worms, we have shown that our model is quite flexible and holds promise for modeling even other kinds of worms. We have compared the results from our model with those obtained from actual worm simulations (Figures 4 and 6), and found the propagation curves to be completely overlapping. This is perfectly understandable, because when a worm permutes the real IP space, every existing structure of the network (like clusters) gets destroyed except the node density ($\frac{V}{N}$) and as a result, the permutation ring gets a uniform distribution of vulnerable hosts. Since the permutation-scanning worm scans on, and jumps to random locations on this ring, its behavior is completely probabilistic and can be fully analyzed. In fact, we expect nothing less than a near-perfect match (with the simulation results) for any model that captures the worm's behavior accurately, as we achieve in this paper.

Our analytical model (and also the worm simulator that we use to verify it) assumes full network connectivity, no delays and no host failures. However, we believe that this model should be effective in modeling most real world worms that are scanning at a very low rate to avoid detection even when such conditions do not always hold true. In our future work, we hope to extend our model to take practical constraints (like congestion, delay etc.) into account so that we can model worms with high scanning rate more effectively.

References

[1] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the Slammer Worm," *In Proc. of IEEE Security and Privacy*, vol. 1, no. 4, pp. 33–39, July 2003.

[2] S. Chen and Y. Tang, "Slowing Down Internet Worms," *Proc. of 24th International Conference on Distributed Computing Systems (ICDCS'04)*, March 2004.

[3] X. Qin, D. Dagon, G. Gu, and a Lee, "Worm Detection Using Local Networks," *Proc. of 20th Annual Computer Security Applications Conf. (ACSAC 2004)*, 2004.

[4] S. Schechter, J. Jung, and A. W. Berger, "Fast Detection of Scanning Worm Infections," *Proc. of Seventh International Symposium on Recent Advances in Intrusion Detection*, September 2004.

[5] N. Weaver, I. Hamadeh, G. Kesidis, and V. Paxson, "Preliminary Results Using Scale-Down to Explore Worm Dynamics," *Proc. of ACM Workshop on Rapid Malcode (WORM)*, March 2004.

[6] Z. Chen, L. Gao, and K. Kwiat, "Modeling the Spread of Active Worms," *Proc. of IEEE INFOCOM'03*, March 2003. [Online]. Available: citeseer.ist.psu.edu/chen03modeling.html

[7] C. C. Zou, W. Gong, and D. Towsley, "Code Red Worm Propagation Modeling and Analysis," *Proc. of 9th ACM Conference on Computer and Communication Security*, pp. 138–147, November 2002. [Online]. Available: citeseer.ist.psu.edu/zou02code.html

[8] S. Staniford, V. Paxson, and N. Weaver, "How to 0wn the Internet in Your Spare Time," *In Proc. of the 11^{th} USENIX Security Symposium*, August 2002.

[9] J. O. Kephart and S. R. White, "Directed-Graph Epidemiological Models of Computer Viruses," *Proc. of 1991 IEEE Symposium on Security and Privacy*, May 1991.

[10] S. Floyd and V. Paxson, "Difficulties in Simulating the Internet," *IEEE/ACM Transactions on Networking*, vol. 9, no. 4, pp. 392–403, 2001.