Inside the Permutation-Scanning Worm: Propagation Modeling and Threat Analysis
Parbati Kumar Manna, Shigang Chen, Member, IEEE, and Sanjay Ranka, Fellow, IEEE

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. Traditionally, most modeling work in this area concentrates on the relatively simple random-scanning worms. However, modeling the permutation-scanning worms, a class of worms that are fast yet stealthy, has been 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 interdependent differential equations, which are then integrated into closed-form solutions that together present the overall propagation 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.

Index Terms—Network Security, Worm, Intrusion

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]–[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 will 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]–[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, 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 simulate a single run of worm propagation for one set of worm/network parameters. 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 on 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 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 cap-
tures 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 then integrate these differential equations to obtain the closed-form solution for propagation. 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; and Section VI gives us the closed-form solutions for the basic permutation-scanning worm. Section VII shows the effects of different worm/network parameters on and real-life network constraints on the worm propagation. Section IX 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 stealthy 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 chosen 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 [1], [3], [4]. 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 hosts will get infected in \( O(\frac{S}{v} \log S) \) 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: 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 \( \text{covered area} \) in Fig. 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 may not have any 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 Fig. 1, which also illustrates other concepts to be defined in this section.

C. Classification of Vulnerable Hosts

In our model, we define classes \( u, i, a, s, x, y, \alpha \) 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.

Below we focus on classifying 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 (Fig. 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 \( \alpha \))**: Those effective hosts that are yet to infect any vulnerable host in their current scanzone (thus have no tail) are termed nascent (class \( \alpha \)). 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-\( \alpha \) 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 is true at the beginning as each of the initially infected hosts belongs to its own scanzone. When a non-\( \alpha \) 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 a non-\( \alpha \) 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.

Fig. 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. Fig. 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. Determining 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 Fig. 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 \( f_{hit} \) 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 \( f_{new}(t) \) (\( f_{old}(t) \)) 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-\( \alpha \) 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

\[
f_{old}(t) = 1 - f_{new}(t) = \frac{x(t)-\alpha(t)}{(1-i(t))+(x(t)-\alpha(t))}.
\]

Answer for Q3: After a newly infected host jumps to a random location to begin its scanning, let \( f_{ineff}(t) \) (\( f_{eff}(t) \)) 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, \( V\alpha(t) \) of them, are located in the covered areas, and there are \( V(x(t)-\alpha(t)) \) tails (single-infection scanzones can be thought of having a covered area of length \( 0 \)). Therefore, \( f_{ineff}(t) = \frac{V\alpha(t)}{V(x(t)-\alpha(t))} \), 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 \( da(t) \), 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.

- \( di(t) \): 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 \( di(t) = x(t) f_{hit} f_{new}(t) \).

- \( dx(t) \): 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 \( x(t) f_{hit} f_{new}(t) V f_{eff}(t) \) new effective hosts after \( dt \). On the other hand, effective hosts hits \( x(t) f_{hit} f_{old}(t) V \) 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 \( dx(t) = x(t) f_{hit} f_{new}(t) f_{eff}(t) - x(t) f_{hit} f_{old}(t) \).

- \( da(t) \): Each nascent host (which is effective by definition) 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 zero, the joint probabilities for two or more hits is negligible. This reduces the number of nascent hosts by \( \alpha(t)V f_{hit} \). 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 \( da(t) = x(t) f_{hit} f_{new}(t) f_{eff}(t) - \alpha(t) f_{hit} \).

- \( dy(t) \): 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 \( x(t) V f_{hit} f_{new}(t) \) 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
\[ dy(t) = x(t)f_{hit}f_{new}(t)f_{eff}(t) - y(t)f_{hit}. \]

- \( ds(t) \): 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 obtain the following equations:

\[ f_{hit} = r \times dt \times \frac{V}{N} \]

\[ f_{old}(t) = \frac{x(t) - \alpha(t)}{1 - i(t) + x(t) - \alpha(t)} \]

\[ f_{new}(t) = \frac{1 - i(t)}{1 - x(t) - \alpha(t)} = 1 - f_{old}(t) \]

\[ f_{eff}(t) = \frac{i(t) - (x(t) - \alpha(t))}{\alpha(t)} \]

\[ f_{eff}(t) = 1 - i(t) + x(t) - \alpha(t) = 1 - f_{eff}(t) \]

\[ di(t) = x(t)f_{hit}f_{new}(t) \]

\[ dx(t) = x(t)f_{hit}f_{new}(t)f_{eff}(t) - x(t)f_{hit}f_{old}(t) \]

\[ do(t) = x(t)f_{hit}f_{new}(t)f_{eff}(t) - \alpha(t)f_{hit} \]

\[ dp(t) = x(t)f_{hit}f_{new}(t)f_{eff}(t) - y(t)f_{hit} \]

\[ ds(t) = x(t)f_{hit}f_{old}(t) + y(t)f_{hit} \]

\[ da(t) = dx(t) + dy(t) \]

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

D. Verification of Our Model

We developed a packet-level simulator for random-scanning
worms and permutation worms whose propagation strategies
are described in Section IV. The simulator is implemented in
C++ with proper encapsulation, i.e., a host object inside the
simulator is not aware of the large picture of the network,
located at different places. In such cases, even if the hosts are sending scanning messages at an identical rate, the effective rate of messages reaching their destination will be different. We also show in Section VIII how to extend our model to other real-life network events like host crashing, patching, quarantining etc.

We juxtapose the propagation graphs from this simulation with ones obtained from our analytical model

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 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.

![State Diagram of a k-jump worm with k=2](image)

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 infections, it retires immediately.

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, any active host (class x, α and y) could have hit anywhere between 0 to k old infections. Therefore, while the terms x(t), α(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 x0, x1, x2 . . . xk−1, xk such that x(t) = ∑k−1j=0 xj(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 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 α2(t). We observe that for calculating the probabilistic figures fold(t), fnew(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 Fig. 5. The transition between the 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 x0(t), α0(t) and y0(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. For example, the number of hosts in a layer increases when hosts in the previous layer hits old infections and move up to the current layer. Similarly, it decreases when hosts in current layer hit old infection and transition into the next layer. Therefore, all the computations for j-layer hosts (where j ≥ 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 fhit, V and N, which are independent of time. For example, fnew denotes fnew(t), dαj denotes dαj(t) and so on. We do not rewrite the equations for fold(t), fnew(t), f_eff(t) and f_ineff(t) since they are the same as in the model for 0-jump worm. ∀j = 0 . . . k, we have

\[
\begin{align*}
dx_j & = \begin{cases}
& \text{if } j = 0, \quad x_{hit} f_{new} f_{eff} - x_j f_{hit} f_{old} ; \\
& \text{if } j > 0, \quad x_{j-1} f_{hit} f_{old} f_{eff} - x_j f_{hit} f_{old} + y_{j-1} f_{hit} f_{eff} ;
\end{cases} \\
dα_j & = \begin{cases}
& \text{if } j = 0, \quad x_{hit} f_{new} f_{eff} - α_j f_{hit} ; \\
& \text{if } j > 0, \quad x_{j-1} f_{hit} f_{old} f_{eff} - α_j f_{hit} + y_{j-1} f_{hit} f_{eff} ;
\end{cases} \\
dy_j & = \begin{cases}
& \text{if } j = 0, \quad x_{hit} f_{new} f_{inff} - y_j f_{hit} ; \\
& \text{if } j > 0, \quad x_{j-1} f_{hit} f_{old} f_{inff} - y_j f_{hit} + y_{j-1} f_{hit} f_{inff} ;
\end{cases}
\end{align*}
\]

Finally, we define the other incremental figures:

\[
\begin{align*}
&dx = \sum_{j=0}^{k} dx_j(t); \quad dy = \sum_{j=0}^{k} dy_j(t); \quad dα = \sum_{j=0}^{k} dα_j(t); \\
&di = x f_{hit} f_{new}; \quad da = dx + dy; \quad ds = x_k f_{hit} f_{old} + y_k f_{hit} ;
\end{align*}
\]
D. Verification of the Correctness of the Model

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 $(s, x_1 \ldots x_k, \alpha, \alpha_0 \ldots \alpha_k, y, y_0 \ldots y_k$ etc.) are zero at $t=0$.

VI. CLOSED FORM SOLUTION FOR THE 0-JUMP WORM

In this section, we condense the set of differential equations into three simple equations that can be further integrated into finding the closed-form solution for infected, active and retired fractions of vulnerable hosts.

A. Infection Speed of 0-jump Worm

We take a conceptual leap to derive the equation for infection speed. In the 0-jump model, a scanning host (say $h_1$) retires after it hits a previously infected host (say $h_2$). The reason for this action is that if $h_1$ continues to scan sequentially beyond $h_2$, it will just be unnecessarily repeating the work of the other scanning host (say $h_3$) that had originally infected $h_2$. However, instead of retiring, if $h_1$ continues to scan (trailing the footsteps of $h_3$) without ever taking a jump upon hitting any previously infected host, then we observe the following for this new “no-retirement” scheme:

- The number of infected hosts $(i(t)V)$ and effective hosts ($x(t)V$) remains unchanged.
- Infection rate, which is directly proportional to the number of effective hosts ($x(t)V$), remains the same.
- For effective hosts, the scanzones remain unchanged.
- Since every covered area belong to some effective host’s scanzone, and neither the total number of effective hosts nor their scanzones does change, the fraction of addresses that fall under the “covered areas” remains unchanged.

Now, under this no-retirement scheme, what will be the relation between $i(t)$ and $x(t)$? We recall that $x(t)$ represents the fraction of the vulnerable host population that can potentially generate new infections. We calculate its value the following way. We first visualize all the $i(t)V$ hosts scanning sequentially on the permutation ring clockwise. Now, at any time point $t$, only those hosts who are currently scanning an address outside a covered area can potentially generate any fresh infection, and the fraction of all such non-covered areas over the total address space $V$ is given by $f_{eff}(t)$ in our original scheme (where a host retires after hitting a previously infected host). Therefore, for our original scheme, we obtain

\[ x(t) = i(t) f_{eff}(t) \]
We had originally derived that the infection rate of the random-jump model is given by \( \frac{di(t)}{dt} = x(t) f_{hit} f_{new}(t) \). By substituting the values of \( x(t) \), \( f_{hit} \) and \( f_{new}(t) \) and rearranging, we obtain

\[
\frac{di(t)}{dt} = \frac{rV}{N} \times \frac{1 - i(t)}{1 - i(t) + x(t) - \alpha(t)}
\]

Thus, \( a(t)V \times r \times dt \times \frac{V}{N} = \) total number of hits = \( di(t) + ds(t) \). In other words, \( ds(t) = a(t)V \times r \times dt \times \frac{V}{N} - di(t) \). Since \( da(t) = di(t) - ds(t) \), it follows that \( da(t) = 2di(t) - a(t)V \times r \times dt \times \frac{V}{N} \). Plugging the value of \( di(t) \) from 2, we obtain the simplified final propagation equations:

\[
\begin{align*}
\frac{di(t)}{dt} &= \frac{rV}{N} \times i(t) \times (1 - i(t)) \\
\frac{da(t)}{dt} &= \frac{rV}{N} \times \left( 2i(t) \times (1 - i(t)) - a(t) \right) \\
\frac{ds(t)}{dt} &= \frac{rV}{N} \times \left( a(t) - i(t) \times (1 - i(t)) \right)
\end{align*}
\]

**B. Condensed Equations for the Propagation Model**

Since each active host has \( \frac{V}{N} \) chance for hitting a vulnerable host for each scan message, in time \( dt \) the expected number of total hits on vulnerable hosts, including both previously uninfected and previously infected, is \( a(t)V \times r \times dt \times \frac{V}{N} \). Hitting a previously uninfected vulnerable host increases \( i(t) \), and hitting a previously infected vulnerable host increases \( s(t) \). Thus, \( a(t)V \times r \times dt \times \frac{V}{N} = \) total number of hits = \( di(t) + ds(t) \). In other words, \( ds(t) = a(t)V \times r \times dt \times \frac{V}{N} - di(t) \). Since \( da(t) = di(t) - ds(t) \), it follows that \( da(t) = 2di(t) - a(t)V \times r \times dt \times \frac{V}{N} \). Plugging the value of \( di(t) \) from 2, we obtain the simplified final propagation equations:

\[
\begin{align*}
\frac{di(t)}{dt} &= \frac{rV}{N} \times i(t) \times (1 - i(t)) \\
\frac{da(t)}{dt} &= \frac{rV}{N} \times \left( 2i(t) \times (1 - i(t)) - a(t) \right) \\
\frac{ds(t)}{dt} &= \frac{rV}{N} \times \left( a(t) - i(t) \times (1 - i(t)) \right)
\end{align*}
\]

C. **Closed-form Solutions for the Propagation Model**

Throughout this paper, we use the notation \( \phi = \frac{V}{r} = a(0) = i(0) = \) fraction of vulnerable host population already infected (and active) at time \( t = 0 \). Using this \( \phi \), we derive the closed form for \( i(t) \), and substituting that value of \( i(t) \) in (4) and integrating, we also obtain the closed-form for \( a(t) \). Finally, \( i(t) = a(t) + s(t) \) yields the closed-form solution for \( s(t) \). Summarizing, we obtain

\[
\begin{align*}
i(t) &= \frac{\phi e^{\frac{rV}{N}t}}{1 - \phi + \phi e^{\frac{rV}{N}t}} - 1 - \phi + \ln(1 - \phi + \phi e^{\frac{rV}{N}t}) + \frac{\phi^2}{2(1 - \phi)} \\
a(t) &= \frac{2(1 - \phi)}{\phi e^{\frac{rV}{N}t}} \left\{ \frac{1 - \phi}{1 - \phi + \phi e^{\frac{rV}{N}t}} - 1 + \phi + \ln(1 - \phi + \phi e^{\frac{rV}{N}t}) + \frac{\phi^2}{2(1 - \phi)} \right\} \\
s(t) &= \frac{\phi e^{\frac{rV}{N}t}}{1 - \phi + \phi e^{\frac{rV}{N}t}} - 1 - \phi + \ln(1 - \phi + \phi e^{\frac{rV}{N}t}) + \frac{\phi^2}{2(1 - \phi)}
\end{align*}
\]

VII. **Usage of the Analytical Model**

In this section, we first describe the benefits of having an analytical model compared to running a simulator. Next, we analyze our model to see what kind of effects 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 will 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 realistic 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 round 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 runtime will be 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 scaling back 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 will a parameter affect the outcome.

B. **Effects of Parameters on Propagation**

Here we analyze the exact effect of each worm/network parameters on the propagation (real-life considerations like congestion and delay will be addressed later in Section VIII).

- **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 \( \phi = \frac{V}{r} \) (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 \( \phi \), which means lower rate of infection initially.
However, the increased probability of getting a hit \( \left( \frac{V}{N} \right) \) 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 \( V \) and \( N \). 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 per our observations from the analytical model, a higher \( v \) simply shifts the time-scale to the right, which implies faster infection. Moreover, a larger hitlist can shorten the initial slow-infection period significantly.

- **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 \( dx(t) \), or \( do(t) \)) 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:** We make an important observation from the figures presented for various values of \( k \) (Fig. 7). We see that with increasing values of \( k \), the slope of the infection curve in Fig. 7 does get steeper, but beyond a certain value of \( k \), 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 \( k \geq 8 \) in our experimental setup, almost all the worms are active when we achieve nearly full infection, which implies a big network footprint. Therefore, it makes little sense to deploy a \( k \)-worm with a high value of \( k \).

**VIII. Practical Consideration**

Although our model was originally conceived assuming ideal conditions (no delay, similar bandwidth, no congestion, no crash etc.), in this section we show that it can easily be extended to take those real-world events into consideration.

**A. Congestion and Bandwidth Variability:**

If for stealth reason the worm sets a small scanning rate \( r \) such as 100 per minute, most infected hosts are likely to have the bandwidth of delivering 100 packets per minute, and our model will be accurate if the deviation caused by Internet delay is negligible. However, if the worm sets its scanning rate \( r \) to be 10,000 per second, then the actual scanning rates of infected hosts may vary due to network congestion. We believe a worm that causes network congestion is not a good worm because it loses stealth (unless its sole purpose is to create headlines by service disruption, which is rarely the case nowadays [12]).

Congestion also happens naturally in the network without worm activity due to the bandwidth limitation and the demand on the routers. As long as the Internet is able to deliver the low scanning rate of most infected hosts, our model can predict the propagation behavior of low-rate stealthy worms. However, we realize that whatever be the reason – processing power of infected host, available bandwidth for the user, congestion of the network – the final result is that on the Internet scale, different hosts are in effect scanning at different rates. Therefore, if we can somehow extend our model to accommodate variable scanning rates from different hosts, we are effectively capturing the real network situation arising out of the reasons mentioned above. Since our model can handle only a fixed scanning rate, we posited that by using average scanning rate, our model should be able to still approximate the variable scanning rate scenario. With that goal in mind, we performed simulations to study the propagation curve of two worms – one whose infected hosts scan at variable rates (Gaussian distribution with a mean value of 10 per time tick and variance of 9), and another with a fixed rate of 10 per time tick. The results are presented in Fig. 8. It shows that the propagation curves of the two worms are very close up to 90% of the infection, after which there is slight discrepancy. Similar results were observed for other variable rate distributions (not presented due to space limitation). Therefore, we argue that our model is indeed able to approximate the propagation of worms by using average scanning rate in real-life scenarios.

**B. Patching and Host Crash:**

Once a vulnerable host gets infected and starts scanning, it may be removed from the vulnerable host pool due to
multiple reasons. For example, upon infection the host may simply crash. Or, the host may get patched after some time. Also, due to scanning activity, an infected host may come under suspicion of the network administrator and resultingly can be taken off the network or quarantined. There can be even other more trivial reasons like the user may simply shut down the host. The result of all these possibilities is that a host is removed from the vulnerable host population. We show that our model can be extended to handle this situation.

We introduce a few additional terms in our model to account for the removal of hosts. First, \( p_q \) denotes the probability of a host being removed every time it scans. Second, \( q(t) \) denotes the number of vulnerable hosts that are removed from the system by time \( t \). As hosts are removed, the vulnerable population also changes; this is why we use \( V(t) \) to indicate the number of hosts at time \( t \) that are actually vulnerable. It is evident that \( V(t) + q(t) = V(0) \) for all \( t \). However, under this “removal” scheme the meaning of \( i(t) \) becomes unclear as some hosts that were infected can now be disinfected. To clear this confusion, we introduce a third new term called \( i_{\text{ever}}(t) \) to denote the fraction of original vulnerable host population \( V(0) \) that were ever infected during the whole propagation, while \( i(t) \) denotes the fraction of \( V(t) \) that are infected. Since \( V(t) \) is not a constant, we rather plot \( i_{\text{ever}}(t) \).

After introducing these new terms, we rewrite the propagation equations of a 0-jump worm the following way:

\[
\begin{align*}
\frac{dx(t)}{dt} &= x(t) f_{\text{hit}}(t) f_{\text{new}}(t) f_{\text{eff}}(t) - x(t) f_{\text{hit}}(t) f_{\text{old}}(t) \\
\frac{d\alpha(t)}{dt} &= x(t) f_{\text{hit}}(t) f_{\text{new}}(t) f_{\text{eff}}(t) - \alpha(t) f_{\text{hit}}(t) - \alpha(t) p_q \\
\frac{dy(t)}{dt} &= x(t) f_{\text{hit}}(t) f_{\text{new}}(t) f_{\text{ineff}}(t) - y(t) f_{\text{hit}}(t) - y(t) p_q \\
\frac{ds(t)}{dt} &= x(t) f_{\text{hit}}(t) f_{\text{old}}(t) + y(t) f_{\text{hit}}(t) \\
\frac{da(t)}{dt} &= dx(t) + dy(t) \\
\frac{dq(t)}{dt} &= x(t) p_q + y(t) p_q \\
\frac{di(t)}{dt} &= x(t) f_{\text{hit}}(t) f_{\text{new}}(t) - dq(t) \\
\frac{di_{\text{ever}}(t)}{dt} &= di(t) + dq(t) \\
\frac{dV(t)}{dt} &= -dq(t)
\end{align*}
\]

Finally, we add the incremental figures like \( i(t+dt) = i(t) + di(t), x(t+dt) = x(t) + dx(t) \) etc. The boundary condition to these set of equations are: \( i(0) = \alpha(0) = x(0) = \phi = \frac{V(0)}{N} \) and \( \alpha(0) = s(0) = y(0) = 0 \), where \( \phi \) is the number of vulnerable hosts in the hitlist \( v \) as a fraction of \( V \) at \( t=0 \). The simulation results are shown in Fig. 9.

C. Internet Delay:

When deriving the propagation model, we implicitly assume that each scan message instantaneously reaches the address being scanned. In reality, the worm will propagate slower due to end-to-end delay of the Internet. Hence, the model in (6) gives an upper bound on the worm’s propagation speed.

In case of a new infection using TCP, it takes one round trip to exchange SYN (which is the scan message) and SYN/ACK, and then it takes a number of round trips to transmit ACK and attack packets. For example, if the worm code size is and each TCP segment is 512 bytes, then under TCP’s slow start it takes three round trips to complete the infection. Internet’s round trip delay rarely exceeds one second [11]. Let \( D \) be a time period that upper-bounds the delay of most infections. Since worm
code is typically short (in order to fit in the call stack without causing the program to crash when buffer-overflow attack is used), \( D \) is expected to be several seconds.

The larger the infection delay is, the slower the worm propagates. Hence, if we artificially set the delay of all infections to the upper bound \( D \) (ignoring the rare cases whose delay exceeds \( D \)), we have a lower bound on the worm propagation speed. It can be shown that this lower bound is simply the propagation curve (6) shifted to the left by \( D \). Combining both the lower bound and the upper bound, we have the following inequality for the actual value of \( i(t) \) after Internet delay is considered. For \( t \geq D \),

\[
\frac{\phi e^{-rV_{N}(t-D)}}{1 - \phi + \phi e^{-rV_{N}(t-D)}} \leq i(t) \leq \frac{\phi e^{-rV_{N}t}}{1 - \phi + \phi e^{-rV_{N}t}} \tag{9}
\]

If a worm wants to stay undetected, it will choose a low scanning rate for better stealthiness (smaller footprint on the Internet) even when that means lower propagation speed and longer propagation time. The propagation time for many known worms can be hours or even tens of minutes to infect the Internet. For these worms, a maximum deviation of several seconds by the model from the reality is relatively small with respect to the much longer overall propagation time. Note that our goal here is not to determine the actual value of \( D \). Instead, we argue that the predictive power of our model is relevant in reality when the Internet delay is small compared to the worm propagation time.

IX. Conclusion

In this paper, we have successfully modeled the propagation characteristics of different varieties of permutation-scanning worms. We have compared the results from our model with those obtained from actual worm simulations (Fig. 4 and 6), and found the propagation curves to be completely overlapping. There is a perfectly understandable reason for this perfect match. When the real IP space is permuted, every existing structures of the network (like clusters) gets destroyed except the node density \( \frac{V_{N}}{N_{n}} \). As a result, the permutation ring gets an even 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 hence can be fully analyzed. As a result, we expect nothing less than a perfect match (with the simulation results) for any model that captures the worm’s behavior accurately, as we achieve in this paper. Finally, though our analytical model was originally conceived assuming ideal network condition, we have shown that it can very well be extended to real-life scenarios like variable bandwidth, congestion and Internet delay, host crash and patching etc.

References


