The Effects of Threading, Infection Time, and Multiple-Attacker Collaboration on Malware Propagation

Yu Zhang, Bharat Bhargava*
Department of Computer Sciences
Purdue University
West Lafayette, USA
Email: zhangyu, bb@cs.purdue.edu

Philipp Hurni
Institute of Computer Science and Applied Mathematics
University of Bern
CH-3012 Bern, Switzerland
Email: hurni@iam.unibe.ch

***Abstract*—Self-propagating malware spreads over the network quickly and automatically. Malware propagation should be modeled accurately for fast detection and defense. State-of-the-art malware propagation models fail to consider a number of issues. First, the malware can scan a host for multiple vulnerabilities on multiple ports. Second, the vulnerability scanning can be done by multiple threads concurrently. Third, the exploitation of vulnerabilities and the infection of vulnerable hosts cannot be done instantly. Fourth, the malware propagation can start from multiple places in the network rather than a single release point. Finally, the malware copies can collaborate with each other to cause much more damage.**

Little was done to understand the effects of Multi-port scanning, Multi-threading, Infection time, Multiple starting points, and Collaboration (MMIMC) on malware propagation. This research quantitatively measures the effects of MMIMC on infected hosts. We employ the Fibonacci Number Sequence (FNS) to model the effects of infection time. We derive the Shift Property, which illustrates that different malware initializations can be represented by shifting their propagations on the time axis. We prove the Linear Property, which shows that the effects of multiple-attacker collaboration can be represented by linear combinations of individual attacks. Experimental results show that the above issues significantly affect malware propagation and verify our analysis.

***Index Terms*—Malware, Thread, Collaboration, Propagation, Network Security.**

I. Introduction

Malware is software designed to compromise computer systems. Examples include Logic Bombs, Viruses, Worms, and Botnets [2], [7]. Malware can be classified into two categories: self-propagating malware and non-self-propagating malware. Self-propagating malware poses a serious threat due to its ability to propagate through networks to infect a large number of hosts. E.g., worms have infected thousands of computers [1], [10], [11], [12]. Malware replicates itself and intrudes vulnerable hosts without human intervention. Malware can carry malicious payloads that can be released upon infection of the vulnerable hosts. Malware can cause significant damages, including consumption of network bandwidth, destructions of infected hosts, and leakage of private information, such as credit card numbers, etc.

Typical Malware propagation consists of a number of steps:
1. *Reconnaissance*: search vulnerable victim hosts by performing port scans;
2. *Infection*: transmit malicious payloads, exploit vulnerabilities on victim hosts to gain control;
3. *Discovery*: perform information-gathering activities on victim hosts, e.g., steal passwords and personal files;
4. *Destruction*: perform destructive activities on victim hosts, e.g., re-format their hard disks.
After the Infection step is done, the malware is ready to propagate from the newly infected host to another one by repeating the whole process. Note that not all malware propagation follow all of the above steps.

To perform a thorough port scan during reconnaissance, malware sends probe packets to each port on each victim host, and analyzes their responses. In a hypothetical scenario, a packet sent to FTP port 21 on a victim host triggers a reply packet, which is then analyzed by malware to infer detailed information, such as the type and version of the operating system, about the victim host. Based on this information, a well-tailored attack can be launched (e.g., exploiting the vulnerability that exists on the particular operating system).

Malware has to perform port scans for a huge number of IP address/port number combinations. In IPv4 networks, the size of the IP address space is 2^{32}, and the size of the port number space is 2^{16}. Hence, the size of the search space for the IP address/port number combination is 2^{48}. While the large size of the search space renders port scanning a daunting task, malware authors have employed sophisticated techniques to perform fast scanning. E.g., many real-world worms search vulnerabilities only on a particular port, which effectively reduces the size of the search space to 2^{32} [9].

It is clear that malware with different scanning and propagation strategies has different propagation time.

*The authors would like to thank Leszek Lilien at West Michigan University for his contributions to this work. The authors would like to thank Ziyu Zhang at Stanford University and Yi Mao at Georgia Institute of Technology for helpful discussions. This work was supported in part by grants from NSF-0242840 and NSF-0219110.

A number of models have been proposed to characterize propagation of worms, including the state-of-the-art Analytical Active Worm Propagation (AAWP) model [1], and the epidemiological two-factor model [10]. Existing malware propagation models fail to consider a number of issues, including the following:

a) *That malware can scan a host for multiple vulnerabilities*: E.g., if malware fails to find any vulnerability on the FTP port 21 of a host, it can look for vulnerabilities on other ports, e.g., the DNS port 53. In case that malware discovers multiple vulnerabilities, it is able to exploit the most promising one according to some criteria (e.g., infection time).

b) *That scanning can be done by multiple threads*: Multi-threaded malware can scan and infect multiple machines concurrently. Moreover, since vulnerabilities exist on many ports, multi-threaded scanning of multiple ports on one host is an effective way to speed up port scans. Most existing models, including the AAWP model, fail to consider that malware may spawn a large number of threads to scan concurrently.

c) *That exploitation of vulnerabilities and infection of victim hosts are not done instantly*: It takes time for malware to transmit its payload, exploit a vulnerability, and subvert the defense system on a victim host. A newly found vulnerable host can neither be infected immediately nor be ready right away to infect other hosts. Although the AAWP model claims to incorporate the infection time, it simply makes the clock ticks larger, without calculating the ratio of scan time to infection/propagation time. In AAWP, all infected hosts perform scanning activities at the next time tick (denote it as t and time tick length as L). Therefore, newly infected hosts that were infected between time (t, t + L) are treated equally: hosts infected near time t perform the same number of scans as those infected near time t + L. Such equal treatment is imprecise. It should be noted that port scans can be done much faster than infections. In the extreme case, Figure 1(c) in [1] assumes that the infection time could be as long as 60 seconds, while the scanning time for one IP/port combination is usually shorter than 0.1 second [16].

Theorem 1 in AAWP is proven by induction on the number of scans. If the scan is successful, it brings in a newly infected host. Hence, each induction step adds at most one host. At the next time tick, the number of infected hosts increases by at most one. AAWP assumes that the scans are performed step by step, i.e., in each step the scanning of one worm is performed, and the number of infected hosts is updated. The assumption differs from most real-world scenarios. For example, the famous NMAP scanner [16] is capable of scanning many hosts in parallel by dividing targets into multiple groups, and scanning an entire group at a time.

d) *That malware propagation can start from multiple places rather than a single starting point, and infected hosts can collaborate to increase damage (e.g., the Botnet [7] and the orchestrated attacks on Estonia [21])*: Multiple attackers can simultaneously release the same malware at multiple places. Researchers suspect that the Witty Worm [2] was released from multiple IP addresses. Malware can be released in different geographical regions as well, e.g., Europe, Asia, and North America, to significantly expedite its propagation. Multiple starting points are not well-represented by existing models.

In summary, little was done to understand the effects of Multi-port scanning, Multi-threading, Infection time, Multiple starting points, and Collaboration (MMIMC) on malware propagation. In this paper, we quantitatively measure the effects of MMIMC on infected hosts. We employ the Fibonacci Number Sequence (FNS) to model the effects of infection time. The extended model can explain the impact of threading, infection time, and multiple-attacker collaboration, as well as the effects of hitlist size, birth rate, and patching rate on malware propagation. We derive the Shift Property, which illustrates that different malware initializations can be represented by shifting their propagations on the time axis. We prove the Linear Property, which shows that the effects of multiple-attacker collaboration can be represented by linear combination of individual attacks. Experimental results show that the above issues significantly affect malware propagation and verify our analysis. To our knowledge, this is the first paper that provides quantitative analysis and experimental results on the effects of MMIMC.

The rest of the paper is organized as follows. In Section II, we discuss related work. In Section III, we quantitatively measure the effects of MMIMC on malware propagation. In Section IV, we conduct experiments to verify our theoretical analysis. Section V concludes this paper.

II. Related Work

Scan Strategy. Over the years, researchers have proposed various scanning algorithms for malware, including: (a) *naive random scanning*, in which malware chooses a random address niformly from the IP address space [1]; (b) *localized scanning*, in which malware scans a local IP address with a high probability p and scans a random address with a low probability (1-p) each time [19]; (c) *importance scanning*, in which malware assumes that the vulnerable hosts are unevenly distributed and such distributions are obtainable [6]; (d) *self-learning scanning*, in which malware estimates the distribution of the vulnerable hosts [18]; (e) *hit-list scanning*, in which malware uses an existing list. e.g., BGP routing table list, social network list, etc., to look for vulnerable hosts [12]; (f) *permutation scanning*, in which malware can determine whether a host is already infected and changes scan targets [12]; (g) *sampling scanning*, in which malware samples a target network before spreading to it [3]; and (h) *passive scanning*, in which malware analyzes the network traffic passively.

Malware Propagation. Wagner et al. [15] present characteristics of worms, including protocol, size of the payload, and scanning strategy, etc. Zou et al. [9] analyze the performances of different propagation strategies. Voyiatzis et al. [14] describe a class of worms that target network components such as routers. Vojnovic et al. [3] discuss how to minimize the required number of scans to infect hosts. Storm Worm [4],

TABLE I
NOTATIONS USED IN THIS PAPER

Notation	Explanation
b	the number of IP addresses on the blacklist of the malware
c	the number of ports scanned for each IP address
w	the number of contagious hosts that can infect other hosts
q	the probability that a given IP address/port combination will be discovered by at least one infected host
d	destruction rate: the number of destructed hosts over the number of infected hosts
k	the number of threads in the malware
p	patching rate: the rate at which the vulnerable machines are patched
r	birth rate: the rate at which the new vulnerable hosts joins the network
v	the number of vulnerable (excluding infected) host/port combinations
V	the number of vulnerable (including infected) host/port combinations
PT	Propagation Time
IT	Infection Time

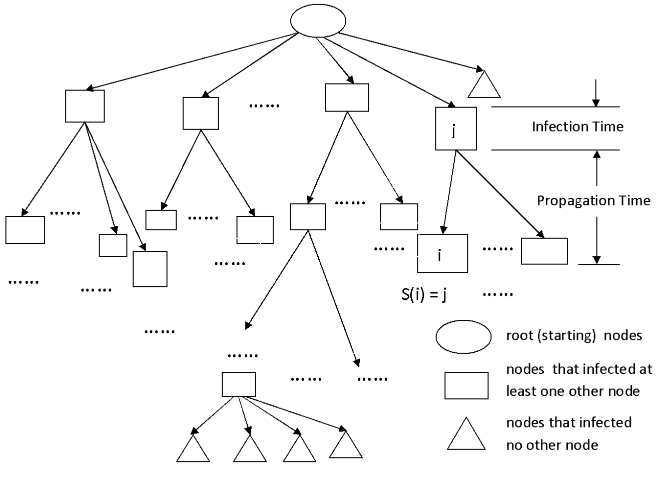
......
j
Infection Time
Propagation Time
i
S(i) = j
root (starting) nodes
nodes that infected at least one other node
nodes that infected no other node

Fig. 1. The malware propagation tree.

[5] uses the Distributed Hash Table (DHT) protocol based on Kademlia [17] to control infected nodes. Chen et al. [1] propose the Analytical Active Worm Propagation (AAWP) model. Zou et al. [10] propose the epidemiological two-factor model. Dagon et al. [23] discuss the taxonomy of Botnets.

III. Analysis of MMIMC and the Generic Fibonacci Malware Propagation (GFMP) Model

In this section, we extend the existing malware propagation models to address the issues of MMIMC.

A. Preliminaries

*1) **Probability on Port Scanning:*** We assume that during the reconnaissance step malware performs port scanning to discover vulnerable ports on the target host.

Malware can scan a part of all IP addresses. For instance, reverse engineering [19], [11] shows that Code Red I and II never scan local (127.0.0.0/8) and multicast (224.0.0.0/8) addresses. This is overlooked by researchers (e.g., in [10] the authors assume that CodeRed scans all IP addresses with equal probability). Assume that IPv4 is in use and malware puts *b* IP addresses on its blacklist, i.e., it never scans those IP addresses. Thus, the number of IP addresses malware scans is (2^{32} - *b*). Assume that malware scans *c* ports for each IP address. The size of the search space for malware is *c*(2^{32} - *b*).

While real-world scanners are mostly multi-threaded [16], existing malware propagation models overlook multi-threading issues. We assume that malware employs multi-threaded programming and scans multiple address/port combinations concurrently. If there are *k* threads for each malware scanning module, we assume that each infected host can scan *k* address/port combinations simultaneously.

We need to calculate how many new vulnerable IP address/port combinations are discovered in each time tick. Note that vulnerability discovery is not equivalent to successful infection. After the vulnerability discovery, malware needs time to propagate to victim hosts and exploit the vulnerability. Assume that there are v_i uninfected vulnerable IP address/port combinations (multiple ports on one host can be infected) at time tick *i* (time steps are equally sized). Given that each infected host can perform *k* scans simultaneously, we can calculate how many out of those v_i combinations can be discovered by all infected hosts.

Denote the number of newly infected hosts as n_i, the number of contagious hosts as *w*, and the probability that a given IP address/port combination is discovered by at least one infected host as *q*. Note $q = \frac{k}{c(2^{32}-b)}$. For a given IP address/port combination, we have:

P(discovered by at least one infected host at time tick i)

$$= 1 - P(not\_discovered\_by\_any\_of\_w\_infected\_hosts)$$
$$= 1 - P(not\_discovered\_by\_one\_infected\_host)^w$$
$$= 1 - (1 - P(discovered\_by\_one\_infected\_host))^w$$
$$= 1 - (1 - q)^w$$

Hence,

$$n_i = [1 - (1 - q)^w]v_i \tag{1}$$

*2) **The Propagation Tree of Self-Propagating Malware:*** Assume that malware propagation starts from a single node. As shown in Fig. 1, the propagation tree of malware $PropT_r$ consists of: (a) a *root node r*: the node where the malware executor releases it; (b) *intermediate nodes*: nodes that caused direct infections of one or more nodes; and (c) *leaf nodes*: nodes that caused no direct infections of other nodes.

Formally, we define:

(1) *Source (Parent) Function S*, such that: S(i) = j, iff node i, j $\in PropT_r$, and j is a parent of i in the tree $PropT_r$. As shown in Fig. 1, if S(i) = j, node i is the child of node j.

(2) *Malware Propagation Tree* $PropT_r$, a directed tree in which each node is either: (a) *root node r*, where $\nexists$ node j $\in PropT_r$ such that j $\neq$ r and S(r) = j; (b) *intermediate node i*, where $\exists$ node j $\in PropT_r$ such that j $\neq$ i and S(j) = i; or (c) *leaf node e*, where $\nexists$ node j $\in PropT_r$ such that j $\neq e$ and S(j) = e, and $\exists$ node k $\in PropT_r$ such that k $\neq$ e and S(e) = k.

*3) **The Propagation Forest of Self-propagating Malware**:* If malware is released at k sources, r_i, i ∈ [1, . . ., k], we can generate one propagation tree for the malware propagation rooted at each source node. The propagation forest of self-propagating malware F_{prop} is the disjoint union of the propagation trees rooted at nodes r_i, i ∈ [1, . . ., k], formally:

$$F_{prop} = \bigcup_{i=1}^{k} PropT_{r_i} \qquad (2)$$

*4) **The Infection Time and Propagation Time**:* As shown in Fig. 1, there is a short delay between the intrusion of the malware and its propagation to other hosts. Such delay includes the time spent on the vulnerability exploitation and subversion of the victim host. We denote the delay as *infection time*.

Intuitively, we define the *Infection Time (IT)* as the time interval between the start of the infection on a particular host (i.e., the time when the host was initially intruded) and the start of propagation on the same host (i.e., the time when the same host was starting to infect other hosts). Formally,

$$IT = T_{StartPropagation} - T_{StartInfection} \qquad (3)$$

Actual infection times may vary and follow particular probability distributions.

We could define the *Propagation Time (PT)* between two hosts as the time interval between the infection of a particular host (denote it as s) and the successful infection of a subsequent target host (denote it as T(s)) that was caused by this particular host. Formally,

$$PT(s, T(s)) = T_{Infection(T(s))} - T_{Infection(s)} \qquad (4)$$

For a *host m* that was never intruded or infected successfully, the time of infection ($T_{Infection(m))}$) is defined as +∞ (infinite).

We can measure the propagation time for all infected hosts and collect statistics about them. E.g., we can calculate the average propagation time. There is one problem with definition (3): it works only if there is at least one subsequent successful infection from the original host (s). If such infection was unsuccessful (e.g., if the target host was invulnerable) or there was no subsequent infection attempt (e.g., if malware on the host was quarantined by administrators) the propagation time is +∞ (infinite).

Alternatively, we can calculate the propagation time from the infected hosts, under the observation that each infected host must be infected by some source host. Hence, we define the *Propagation Time (PT)* between a host and its infector as: the time interval between the successful infection of a particular host (denote it as t) and the infection of the host that infected t [8] (denote it as S(t)). Formally,

$$PT(S(t), t) = T_{Infection(t)} - T_{Infection(S(t))} \qquad (5)$$

B. Generic Fibonacci Malware Propagation (GFMP) Model

We employ Fibonacci Number Sequence (FNS) to model infection time. Recall that in the Fibonacci rabbit problem, newly-born rabbits cannot give birth to baby rabbits immediately. Instead, they need some time to mature, which is reminiscent of the infection/propagation time problem discussed above: a captured host cannot scan and infect other hosts until its infection matures, i.e., until it is completely infected.

1) Recursive Equation for the Malware Propagation

Denote the number of all vulnerable hosts in the beginning as *V* and the number of infected hosts as I_j, where j denotes the time tick. Denote the length of the time slice between time ticks as *L* (one time slice could represent one second). Assume that the administrators may patch the vulnerable hosts. Assume that the propagation time is two time slices for all infections. Hence, the newly infected hosts intruded at time t are not able to infect new hosts at time *t* + *L*, but will be able to infect new hosts at time *t* + 2*L*. At time tick *j* + 2, there are I_j infected hosts that are contagious and can infect other hosts. Formally:

$$w = I_j \qquad (6)$$

At time tick *j* + 1, the number of uninfected vulnerable hosts is the number of all unpatched vulnerable (including infected and newly born) hosts minus the number of infected vulnerable hosts. Assume that malware can carry destructive payloads (e.g., programs that can re-format the hard drive). In this case, the destructed hosts are wiped out and removed from the vulnerable host list. Note that neither dead (or significantly damaged) nor newly-born hosts could be patched.

We define *destruction rate* as the number of destructed hosts divided by the number of infected hosts, considering that only infected hosts can be destroyed. Therefore, we calculate the number of dead hosts by multiplying the destruction rate by the number of infected hosts, instead of the number of all vulnerable hosts. We denote the destruction rate of the hosts as *d*, the birth rate of the vulnerable hosts (e.g., new vulnerable hosts that just joined the network) as *r*, and the patching rate of infected hosts as *p*. Formally, the number of hosts that are vulnerable (including infected and newly-born) and can be patched at time tick *j* + 1 is:

$$v_{j+1} = (1-p)v_j - dI_j + rv_j = (1-p+r)v_j - dI_j$$

This is a recursive equation. We expand the recursion and get:

$$v_{j+1} = (1-p+r)^{j+1}v_0 - \sum_{k=0}^{j}(1-p+r)^k dI_{j-k}.$$

Given that v_0 = V, we have:

$$v_{j+1} = (1-p+r)^{j+1}V - \sum_{k=0}^{j}(1-p+r)^k dI_{j-k} \qquad (7)$$

The number of hosts that are vulnerable but not infected is:

$$v'_{j+1} = v_{j+1} - I_{j+1} \qquad (8)$$

After one time slice (time tick *j* + 2), without considering destruction and patching, the number of infected hosts is the sum of the number of infected hosts at the previous time tick (j + 1) and the number of newly infected hosts during the time slice. The number of infected hosts that died or were patched during the time slice is

$$dp_{j+1} = (d+p)I_{j+1} \tag{9}$$

The number of newly infected hosts is calculated in Section III-A.

Given (1), (6), (7), (8), (9), we have:

$$\begin{aligned} & I_{j+2} \\ & = I_{j+1} + n_{j+1} - dp_{j+1} \\ & = I_{j+1} + v'_{j+1}(1-(1-q)^{I_j}) - (d+p)I_{j+1} \\ & = (1-d-p)I_{j+1} + [(1-p+r)^{j+1}V - \\ & \sum_{k=0}^{j}((1-p+r)^k dI_{j-k}) - I_{j+1}][1-(1-q)^{I_j}] \end{aligned} \tag{10}$$

Note this recursive growth function applies when there is at least one vulnerable host.

*2) **Special Cases***

Special cases are as follows:

a) If the birth and patching rates are equal, (10) can be simplified to:

$$\begin{aligned} & I_{j+2} \\ & = (1-d-p)I_{j+1} + \\ & (V - d\sum_{k=0}^{j} I_{j-k} - I_{j+1})[1-(1-q)^{I_j}] \end{aligned} \tag{11}$$

b) If the birth, destruction, and patching rates are all zero, (10) can be simplified to:

$$I_{j+2} = I_{j+1} + (V - I_{j+1})[1-(1-q)^{I_j}] \tag{12}$$

c) Binomial expansion can be used to expand and simplify 1 - (1 - q)^{I_j}:

$$\begin{aligned} & 1-(1-q)^{I_j} \\ & = 1 - \sum_{m=0}^{I_j} \binom{I_j}{m} (-q)^m \end{aligned} \tag{13}$$

We observe that: k represents the multi-threading level of the malware propagation scanner, and normally ranges from 1 to 2^{10} or one thousand; V represents the number of vulnerable hosts(including infected hosts), and is normally smaller than 2^{20} or one million; c represents the number of ports that the malware is scanning, and $c > 0$; and b represents the number of IP addresses that the malware puts on the blacklist. If the malware puts local and multicast addresses on the blacklist only, 2^{32} - b $\approx$ 2^{32}. Hence, qV = $\frac{kV}{c(2^{32}-b)}$ $<$ $\frac{2^{10}\times 2^{20}}{2^{32}}$ = $\frac{1}{4}$. Note that these are conservative estimations since normally k is much smaller than 2^{10} and V is smaller than 2^{20}. Since the number of infected hosts cannot be larger than the number of all vulnerable hosts, i.e., $I_j \leq$ V, we conclude that qI_j is small. Therefore, we can safely discard the high order elements in Equation 13. We can rewrite Equation (12) as:

$$\begin{aligned} I_{j+2} & = I_{j+1} + (V - I_{j+1})[1-(1-q)^{I_j}] \\ & = I_{j+1} + (V - I_{j+1})[1 - \sum_{m=0}^{1} \binom{I_j}{m} (-q)^m] \\ & = I_{j+1} + (V - I_{j+1}) \binom{I_j}{1} (q) \\ & = I_{j+1} + qI_j(V - I_{j+1}) \\ & = I_{j+1} + qI_jV - qI_jI_{j+1} \end{aligned}$$

During the initial phase of the spread of the malware, $\frac{I_{j+1}}{V}$ is a small number, so we can safely throw away $-qI_jI_{j+1}$. Therefore:

$$I_{j+2} = I_{j+1} + qVI_j \tag{14}$$

Equation 14 suggests that the initial spread of the malware approximately follows the Generic Lucas Number Sequence (LNS) [20] with α = 1 and β = -qV:

$$I_j = \begin{cases} x & \text{if j = 0;} \\ y & \text{if j = 1;} \\ I_{j-1} - (-qV)I_{j-2} & \text{if j > 1.} \end{cases} \tag{15}$$

d) We now discuss the effects of different lengths of the propagation time. Equation 14 holds when the propagation time is $2L$ (twice as much as the length of the unit time slice). Generally, if propagation time is eL, where e is an integer, we have:

$$I_{j+2} = I_{j+1} + qVI_{j+2-e}, \quad where \quad j+2 > e \tag{16}$$

We now have the equation that quantitatively measure the effects of propagation time. The equation shows that the longer propagation time is, the slower the malware propagates, which follows the intuition that longer propagation time hampers malicious activities of newly infected hosts.

C. Properties of the GFMP Model

We use the GFMP model to study the issues of threading, infection time, multiple starting points, and collaborations. Due to space limitations, we omit the discussion of properties of FNS and use them directly. Interested readers are referred to [13] for details.

*1) **Multi-threading and the Closed-Form Expression**:* The closed-form expression for the number of infected hosts at time tick j, when x = 0 and y = 1, is:

$$\begin{aligned} I_j & = \frac{\phi^j}{\theta}, \quad where \quad \theta = \sqrt{1+4qV} \quad and \quad \phi = \frac{1+\theta}{2} \\ & = \frac{[\sqrt{c(2^{32}-b)} + \sqrt{c(2^{32}-b)+4kV}]^j}{2\sqrt{c(2^{32}-b)+4kV}[2\sqrt{c(2^{32}-b)}]^{j-1}} \end{aligned}$$

Note that the malware propagation stops when all vulnerable hosts that can be infected are infected. Hence, during the propagation $I_j \leq V$. Hence, we can rewrite I_j as:

$$I_j = \begin{cases} \lambda, & \text{if } \lambda \leq \text{V;} \\ V, & \text{if } \lambda > \text{V.} \\ & (\lambda = \frac{[\sqrt{c(2^{32}-b)}+\sqrt{c(2^{32}-b)+4kV}]^j}{2\sqrt{c(2^{32}-b)+4kV}[2\sqrt{c(2^{32}-b)}]^{j-1}}) \end{cases} \tag{17}$$

In Equation 17, *k* denotes the number of active threads in the malware scanner. As *k* increases, the infection rate increases. However, note that multi-threaded programs can easily generate huge network traffic by sending out a large number of packets. While context switching for threads are smaller than those of processes, the costs increase as *k* increases. Real-world multi-threaded malware normally employs 10 - 100 threads. Equation 17 was derived from Equation 14, where we assume that the number of previously infected hosts is much smaller than the number of all vulnerable hosts ($\frac{I_{j-1}}{V}$ is small), and dropped $-\frac{k}{c(2^{32}-b)}I_jI_{j+1}$. Hence, Equation (17) grows faster than actual malware propagation when the number of infected hosts is large. Experimental results that support Equation 17 are discussed in Section IV.

*2) **Sophisticated Scanning and the Shift Property**:* In Section III-C1, we derived the closed-form expression when malware employs multi-threaded random scanning, and the initial values of *x* and *y* are 0 and 1, respectively. However, malware can employ more sophisticated scanning techniques, such as a combination of scanning strategies. Malware can use hitlist scanning to infect a large number of pre-selected vulnerable hosts [12] before performing regular random scanning on newly infected hosts. Our extended model represents such scanning strategies by different initializations of *x* and *y*. E.g., if the size of the hitlist is *h*, we assume that at time tick 1 the number of infected hosts is *h* (the original release point of malware) instead of 1, i.e., *x* = 0 and *y* = *h*.

According to the Shift Property of FNS, The Generic LNS sequence determined by Equation 17 with initial values *x* and *y* can be calculated as:

$$GI_{x,y,j} = I_{[j+\frac{\log(y\phi+x)}{\log\phi}-1]} \tag{18}$$

When *x* = 0 and *y* = h, we have:

$$GI_{0,h,j} = I_{[j+\frac{\log(h\phi)}{\log\phi}-1]} \tag{19}$$

Hence, the number of infected hosts of the malware with a hitlist of size *h* and the combined scanning strategy at time *j* can be represented by the number of infected hosts of the original random-scanning malware at time (*j* + *s*), where *s* is the shifting number $\frac{\log(h\phi)}{\log\phi} - 1$.

Furthermore, according to properties of the FNS,

$$GI_{x,y,j} = xI_{j-1} + yI_j$$

Hence, the propagation of malware employing the combined hitlist and random scanning is the linear combination of two propagations of malware employing random scanning only. When *x* = 0 and *y* = *h*, we have:

$$GI_{0,h,j} = hI_j \tag{20}$$

We call *h* the linear Fibonacci Coefficient (FC) of the linear combination.

*3) **Multiple Starting Points, Collaborative Attacks and the Linear Property**:* Malware propagation can start from multiple places in the network rather than from a single point, and infected hosts can collaborate with each other to cause much more damage. E.g., the coordinated Botnet zombie nodes can collaborate to launch DoS attacks [7], and the well-orchestrated collaborative attacks on Estonia caused large-scale disruptions [21].

We consider the representation of the following attacks:

Case 1. There are *m* uncoordinated attackers who release the same copy of malware at *m* places simultaneously. We assume that malware employs the localized random scanning strategy. We assume that the search spaces of attackers are independent (e.g., attackers divide the whole IP address space equally into *m* parts and each attacker will be responsible for one part). For the initializations, we assume that *x* = 0 and *y* = 1 for all attackers. According to Equation 15, the propagation of malware released by all attackers can be represented as $I_{0,1,j}$ because their initial values and β coefficients are the same. Note that $|\beta|$ = $\frac{kV}{c(\frac{2^{32}}{m}-b)}$ now since the search space for each attacker is now reduced to $\frac{2^{32}}{m}$. Recall that we have $|\beta| < \frac{1}{4}$. As discussed in Section III-B, if we assume that *V* = 2^{20} and *b* = 0, we have $\frac{kV}{c(\frac{2^{32}}{m}-b)} = \frac{mk}{2^{12}c} < \frac{1}{4}$. Hence, $\frac{mk}{c} < 2^{10}$, which means that the product of the number of threads per malware and the number of attackers divided by the number of scanned ports is smaller than 1024, if there are one million vulnerable hosts. We assume that this condition holds and denote the propagation of the whole collaborative attack as $ITOTAL_{x_{total},y_{total},j}$.

According to the Linear Property of the FNS, we have:

$$\begin{aligned} ITOTAL_{x_{total},y_{total},j} &= \sum_{n=0}^{m-1} I_{0,1,j} \\ &= mI_{0,1,j} \\ &= I_{0,m,j} \end{aligned}$$

Hence, the number of infected hosts of *m* uncoordinated attacks that perform localized scanning is equivalent to that of the single attack released at one point with initial values x_{total} = 0, and y_{total} = *m*.

Case 2. There are still *m* collaborative attackers releasing malware. We assume that malware employs the sophisticated scanning strategy (but each malware copy shares the same search space) and malware at different hosts can communicate with each other to avoid duplicate infection attempts. Note we do not assume that infected hosts can avoid duplicate scanning (in which multiple attackers can be modeled as one attacker with a huge number of threads and minimal thread maintenance costs). We assume that initial values of the propagation of the malware released by Attacker A_n are x_n and y_n (n∈[0, . . ., m)). We still denote the propagation of the whole collaborative attack as $ITOTAL_{x_{total},y_{total},j}$.

According to Linear Property of the FNS, we have:

$$\begin{aligned} ITOTAL_{x_{total},y_{total},j} &= \sum_{n=0}^{m-1} I_{x_n,y_n,j} \\ &= I_{\sum_{n=0}^{1} x_n, \sum_{n=0}^{1} y_n, j} + \sum_{n=2}^{m-1} I_{x_n,y_n,j} \\ &= \dots \\ &= I_{\sum_{n=0}^{m-1} x_n, \sum_{n=0}^{m-1} y_n, j} \end{aligned} \tag{21}$$

Hence, the power of the *m* collaborative attacks is equivalent to the single attack released at one point with initial values x_{total} = $\sum_{n=0}^{m-1} x_n$, and y_{total} = $\sum_{n=0}^{m-1} y_n$. Equation 21 quantifies the power of collaborative attacks, and grows much faster than Equation 15.

IV. Experiments

In this section, we present the experimental results on the impact of threading, infection time, and multiple-attacker collaboration, as well as the effects of hitlist size, birth rate, and patching rate on malware propagation. We have conducted the experiments on a network that consists of a Pentium 4 workstation and virtual machines. We simulate the worm propagation and use one machine to simulate multiple victim hosts. We implemented the malware propagation model in C++. Without loss of generality, in all the experiments, we set *V* (the number of all vulnerable hosts) to 1,000,000, *c* (the number of ports the malware scans for one host) to 1, and *b* (the number of IP addresses that the malware does not scan) to the size of local and multicast address space, which is approximately 2^{25}.

A. Verification of the GFMP model: the Shift Property

We perform experiments to verify our theoretical GFMP model before employing it to study the effects of other parameters. In particular, we want to show the Shift Property discussed in Section III-C2. In this experiment, we set *k* to 100, *d* to 0, *p* to 0.0002, and *r* to 0.0002.

From Equation 20, $GI_{0,h,j} = hI_j$, the number of infected hosts with hitlist size *h* divided by the number of infected hosts with hitlist size 1 is *h*. Hence, if sizes of the hitlists are 2, 100, and 200, the quotients are 2, 100, and 200, respectively. Note that the number of infected hosts with hitlist size 200 is $\frac{200}{100}$=2 times of the number of infected hosts with hitlist size 100.

Fig. 2 shows the results on the propagation with hitlist sizes 1, 2, 100, and 200. Note that numbers of infected hosts for hitlist sizes 1 and 2 are enlarged 100 times. The results confirm our theoretical analysis, and verify the Shift Property. For instance, numbers of infected hosts with hitlist size 100 (or 200) are essentially coincident with the numbers of infected hosts (enlarged 100 times) with hitlist size 1 (or 2). Numbers of infected hosts with hitlist size 200 are approximately twice as many as those with hitlist size 100.

According to Equation 19, $GI_{0,h,j} = I_{[j+\frac{\log(h\phi)}{\log\phi}-1]}$. Therefore, we can compute the number of shifts required to calculate the number of infected hosts with hitlist size h = 100. Since *k* = 100, *V* = 2^{20}, *b* = 2^{25}, *c* = 1, we have: $q = \frac{100*2^{20}}{1*(2^{32}-2^{25})}$ = $\frac{100}{32*127}$ = 0.025. Hence, θ = $\sqrt{1+4q}$ = $\sqrt{1.098}$ = 1.048, and ϕ = $\frac{1+\theta}{2}$ = 1.024 Therefore, the number of shifts is : $\frac{\log((h)\phi)}{\log\phi}-1$ = $\frac{log(100*\phi)}{log\phi}$ -1 = $\frac{2.010}{0.010}$ - 1 = 200.

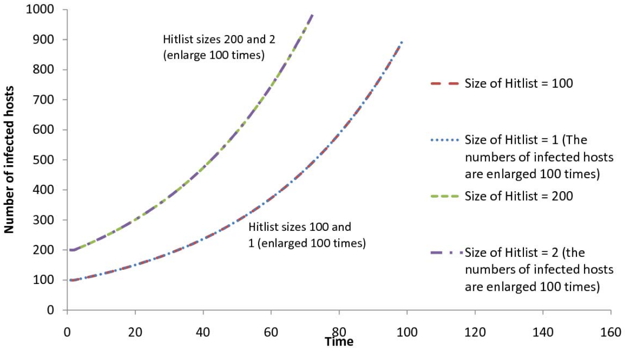

Fig. 2. The propagation of hitlist size 100 and 200, and 100 times of hitlist sizes 1 and 2.

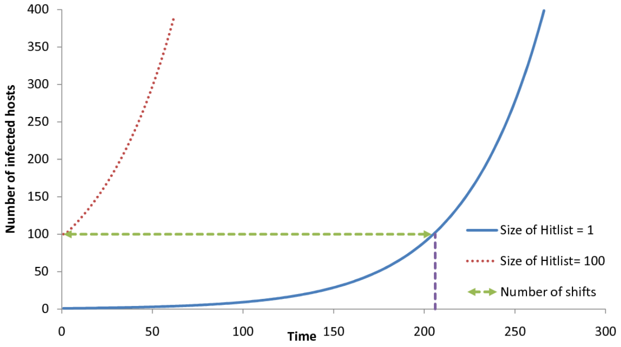

Fig. 3. The propagation of hitlist size 100 and 200, and 100 times of hitlist sizes 1 and 2.

Fig. 3 confirms our theoretical analysis. The solid line shows the propagation with hitlist size 1. The dotted line shows the propagation with hitlist size 100. The dashed line that connects the solid and dotted lines illustrates the number of required shifts, which is approximately constant. The projection of the dashed line on the x-axis shows that the number of shifts is roughly equal to 200, which verifies our analytical result.

B. The effect of different hitlist sizes on the multi-threaded propagation

We have conducted experiments to evaluate whether the hitlist scanning can accelerate the propagation of multi-threaded malware, and compared the effects of different hitlist sizes on the malware propagation. In this experiment, we set *k* (the number of threads in the malware vulnerability scanner of the malware) to 100, *d* (destruction rate) to 0.0001, *p* (patching rate) to 0.0002, and *r* (birth rate) to 0. Note that we set birth rate to 0 to show the effects of threading (otherwise the increased number of infected hosts might be caused by newly joined vulnerable hosts).

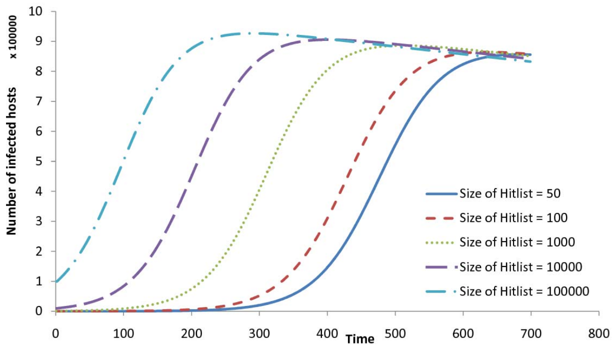

Fig. 4. The Propagation of the multi-threaded malware with different hitlist sizes.

Fig. 4 shows the malware propagation with hitlist sizes 50, 100, 1,000, 10,000, and 100,000. We observe that the propagation speed of the multi-threaded malware increases as the size of the hitlist increases. Specifically, with hitlists of sizes 100,000, 10,000, 1,000, 100, and 50, the malware propagated to 500,000 hosts in 100, 210, 319, 441, and 489 time ticks (seconds), respectively. We conclude that the hitlist scanning can effectively accelerate the multi-threaded malware propagation, especially when the size of the hitlist is large.

When the size of the hitlist is 100,000, the malware propagation reached its peak after 290 time ticks, after which the actual number of infected hosts decreased. Such decrease is caused by patching and destruction. In our experiment, destruction and patching rates are not zero. Moreover, we set the birth rate to zero so that no new vulnerable hosts will join the network. Therefore, after the malware propagation reached its peak, there will be no new vulnerable host to infect, and patched hosts can no longer be infected. Hence, the number of infected hosts decreases. Note that different birth and patching rates can cause different propagation behavior. We discuss our experimental results on the birth patching rates in Sections IV-D and IV-E, respectively.

C. The effect of different threading-levels

We conducted experiments to study how the number of scanning threads affects malware propagation. In this experiment, we set d to 0.0001, p to 0.0002, r to 0, and the hitlist size to 10000.

Fig. 5 shows the propagation with different threading-levels: 50, 100, and 250. We observe that the propagation speed increases as the number of threads in the malware increases. The effects of the multi-threads are significant: when the number of threads is 50, the malware took almost 700 time ticks to infect 800,000 hosts, while the same malware took approximately 300 time ticks with 100 threads and 100 time ticks with 250 threads to accomplish the same task. Note that when the number of threads is 250, the malware propagation reaches its peak in less than 200 time ticks. The number of infected hosts then decreased because of destruction and patching, since we assume that a patched host cannot be infected again in this experiment. The patching rate we set in this experiment is fairly high (0.0002), which means that in every time tick two out of one thousand infected hosts are patched.

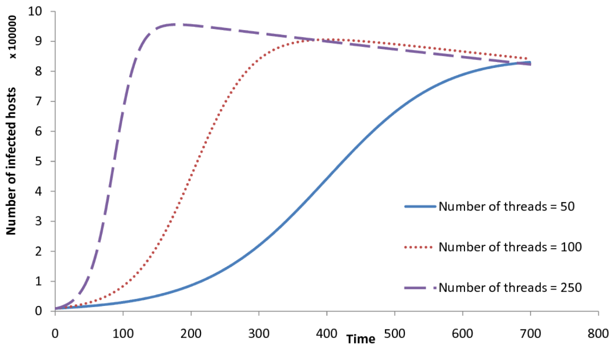

Fig. 5. The malware propagation with different number of threads.

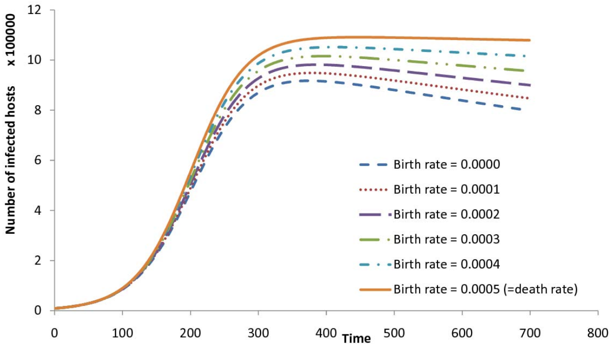

Fig. 6. The malware propagation with different birth rates.

D. The effect of different birth rates

We conducted experiments to study the effects of different birth rates on the malware propagation. In this experiment, we set d to 0.0005, p to 0.0000, k to 100, and the hitlist size to 10,000. Note we set the patching rate to 0 to focus on the birth rate.

Fig. 6 shows the propagation with birth rates 0, 0.0001, 0.0002, 0.0003, 0.0004, and 0.0005. Note that when the birth rate is 0.0005, it is equal to the destruction rate (0.0005). We observe that the birth rate does matter during the malware propagation. Specifically, we observe that the number of infected hosts peaked at 1,050,000 when the birth rate is 0.0004, while the number of the infected hosts peaked at only 917,000 when the birth rate is 0.

Moreover, when the birth rate is 0.0005, which is equal to the destruction rate, we observe that the number of infected hosts peaked at 1,080,000. The number of infected hosts neither increased nor decreased afterwards. Therefore, an equilibrium was reached: although 5 out of 10,000 hosts are destructed in each time slice, 5 out of 10,000 hosts are newly born and infected by the malware in each time slice.

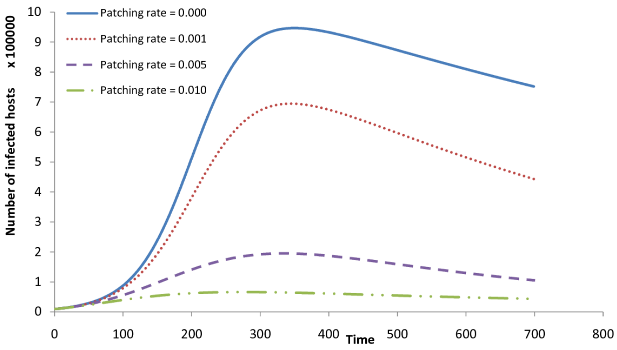

Fig. 7. The malware propagation with different patching rates.

E. The effect of different patching rates

We performed experiments to evaluate the effects of different patching rates on the malware propagation. In this experiment, we set d to 0.0001, r to 0.0003, k to 100, and the hitlist size to 10,000.

Fig. 7 shows the malware propagation with patching rates ranging from 0 to 0.010. We observe that patching significantly reduces the number of infected hosts. Specifically, we observe that the malware propagation peaked at 900,000 hosts when there was no patching, and the number of hosts dropped to approximately 700,000 when the patching rate was just 0.001, which means that only one out of one thousand hosts is patched. The malware propagation peaked at only 193,000 hosts when the patching rate was 0.005, and the malware propagation was significantly reduced and peaked at only 66,000 hosts when the patching rate was 0.01 (one out of one hundred hosts). Therefore, we conclude that patching can significantly diminish the malware propagation and should be employed in all networks.

F. The effect of multiple attackers

We conducted experiments to study the effects of multiple attackers on the malware propagation. In this experiment, we set d to 0.005, p to 0.005, r to 0.03, and k to 100. We first set the hitlist size to 100 and 200, respectively, and performed the experiments. Then, we simulated the multiple-attack scenario discussed in Case 2 of Section III-C3: there are two collaborative attackers, one with hitlist size 100, and the other with hitlist size 200. The two attackers start at the same time, and communicate with each other to avoid duplicate infection attempts.

According to Equation 21, the effects of a collaborative attack is the sum of the individual attacks. Fig. 8 presents the experimental results. The dotted line represents the propagation with hitlist size 100. The dashed line represents the propagation with hitlist size 200. The solid line represents the propagation with two attackers, one with hitlist size 100 and the other with hitlist size 200. The number of infected hosts for the collaborative attack is approximately the sum of the number of hosts infected for the individual attacks with hitlist sizes 100 and 200 initially, which confirms Equation 21.

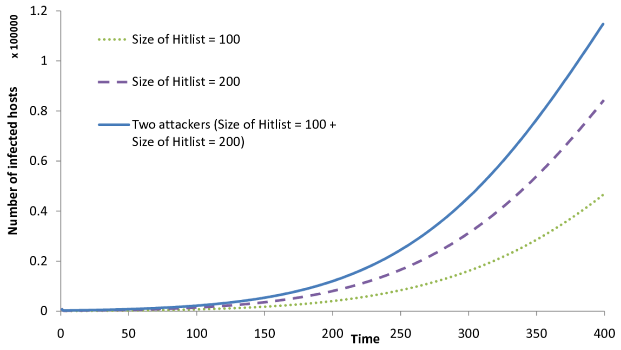

Fig. 8. The malware propagation with multiple attackers.

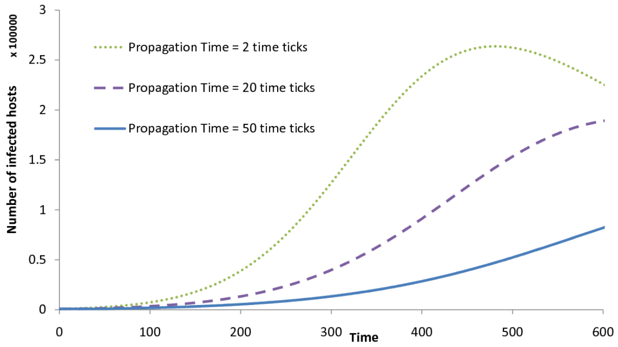

Fig. 9. The malware propagation with different propagation times.

However, we note that after around time tick 300, the sum of the number of infected hosts for the individual attacks become larger than the number of infected hosts for the collaborative attacks, which means:

$$ITOTAL_{x_{total},y_{total},j} < I_{\sum_{n=0}^{m-1} x_n,\sum_{n=0}^{m-1} y_n,j}$$

The explanation is that the number of infected hosts for the collaborative attacks increases more slowly due to the contention between the collaborating attackers. In Case 2 of Section III-C3, we assume that the collaborative attackers can avoid duplicate infection attempts, but cannot avoid duplicate scanning. Hence, some scanning activities in the collaborative attack may collide and such collision can decrease the efficiency of the collaborative attack.

G. The effect of different propagation times

We performed experiments to evaluate the effects of different propagation times on the malware propagation. In this experiment, we set d to 0.005, p to 0.005, r to 0.003, k to 100, and the hitlist size to 1,000. Note that these destruction and patching rates are high. The sum of the two rates is 0.005 + 0.005 = 0.01. Note that the birth rate is 0.003, which is smaller than the patching rate.

Fig. 9 shows the malware propagation with different propagation times: 2 time slices, 20 time slices, and 50 time slices. We observe that as the propagation time increases, the propagation speed decreases. In 200 time slices, the malware infected 38,416, 13,249, and 5,512 hosts, with the 2-time-slice, the 20-time-slice, and the 50-time-slice propagation times, respectively. In 300 time slices, the malware infected 125,980, 39,497, and 13,403 hosts, with the 2-time-slice, the 20-time-slice, and the 50-time-slice propagation times, respectively. Furthermore, the malware propagation reached its peak at time tick 482 with 263,732 infected hosts with the 2-time-slice propagation time, while the malware propagation with the 20-time-slice and the 50-time-slice propagation times are still in the process of trying to infect more nodes. Therefore, we conclude that the defenders fighting malware should try to maximize its propagation time.

H. Comparison with existing models

In this subsection, we show that our model is better by comparing it to existing models. Fig. 1 in [1] shows the results obtained by the AAWP model. Their experiments employ one million vulnerable machines, a scanning rate of 100 scans/second, a death rate of 0.001/second, and random scanning.

The leftmost graph in Fig. 1 in [1] illustrates the effects of hitlist size. In comparison, our results (Fig. 2, 3, and 4) verify the important shift property, and measure the impact of multi-threading and hitlist size on malware propagation. Our results are more practical since threading is employed by most real-world malware. The middle graph in Fig. 1 in [1] illustrates the effects of patching rate. In comparison, our results (Fig. 6 and 7) provide more insights into the effects of patching/birth/death rates. We incorporate not only patching and death rates in our studies, but also the birth rate. Our results on the birth rate (Fig. 6) are significant, especially for wireless and peer-to-peer networks, in which hosts may join or leave at any time. Our results show that death rate can cause the number of infected hosts to decrease over time, which cannot be inferred easily from Fig. 1 in [1].

The rightmost graph in Fig. 1 in [1] illustrates the impact of infection time. While their results show that infection times do not affect malware propagation significantly, our results show otherwise. Our explanation is that we consider the impact of threading, infection time, and multiple-attacker collaboration, as well as the effects of hitlist size, birth rate, and patching rate on malware propagation. Our results are more intuitive because modern malware has very high propagation speed and longer infection time leaves less vulnerable hosts infected.

In all, the comparison shows that our model is more accurate and complete by considering the issues of MMIMC, and provides more insights into malware propagation.

V. Conclusion

In this paper, we quantitatively study issues of Multi-port scanning, Multi-threading, Infection time, Multiple starting points, and Collaboration (MMIMC) in malware propagation. To our knowledge, there is no previous study on the effects of MMIMC. We discuss the limitations of current models, and explain the impact of threading, infection time, and collaboration, as well as the effects of hitlist size, birth rate, and patching rate. We consider the multi-threading issue during the calculation of probability of successful scans. We model the infection time and propagation time of the malware by employing Fibonacci Number Sequence. We derive the Shift Property and the Linear Property. We theoretically analyze the effects of the above issues, and perform experiments to verify the theoretical results.

In Section III, we assume that propagation time is the same for all infections. In the real world, propagation time for different infections may vary. If the propagation time is three time slices, we can apply the Tribonacci Number Sequence [22] to study the malware propagation. Analysis of effects of varying propagation times is the subject for future work.

References

[1] Z. Chen, L. Gao, and K. Kwiat, Modeling the Spread of Active Worms, Proc. of the IEEE INFOCOM, Apr. 2003

[2] http://www.caida.org/research/security/witty/, last accessed *Jul. 1, 2009*

[3] M. Vojnovic, V. Gupta, T. Karagiannis, and C. Gkantsidis, Sampling Strategies for Epidemic-Style Information Dissemination, Proc. of the IEEE INFOCOM, Apr. 2008

[4] S. Sarat and A. Terzis, Measuring the Storm Worm Network. Technical Report 01-10-2007, http://hinrg.cs.jhu.edu/uploads/Main/STORMTR.pdf

[5] C. Kanich, K. Levchenko, and B. Enright, G. M.Voelker and S. Savage, The Heisenbot Uncertainty Problem: Challenges in Separating Bots from Chaff, Proc. of the USENIX Workshop on Large-Scale Exploits and Emergent Threats, San Franciso, CA, Apr. 2008

[6] Z. Chen and C. Ji, Optimal Worm-Scanning Method Using Vulnerable-Host Distributions International Journal of Security and Networks, Special Issue on Computer and Network Security, Vol. 2, 2007

[7] R. Vogt, J. Aycock, and M. Jacobson, Army of Botnets, Proc. of the Network and Distributed System Security Symposium, San Diego, CA, Feb. 2007

[8] A. Svensson, A note on generation times in epidemic models, Mathematical Biosciences, Vol. 208, Iss. 1, Jul. 2007

[9] C. Zou, D. Towsley, and W. Gong, On the Performance of Internet Worm Scanning Strategies, Performance Evaluation, Jul. 2006

[10] C. Zou, W. Gong, and D. Towsley, Code Red Worm Propagation Modeling and Analysis, Proc. of 9th ACM Conference on Computer and Communication Security , Washington D.C., Nov. 2002

[11] D. Moore, C. Shannon, and J. Brown, Code-Red: a case study on the spread and victims of an Internet Worm. In Proc. ACM/USENIX Internet Measurement Workshop, France, Nov. 2002

[12] S. Staniford, V. Paxson and N. Weaver, How to Own the Internet in Your Spare Time, Proc. of the 11th USENIX Security Symposium, Aug. 2002

[13] Y. Zhang and B.Bhargava, Fibonacci Modeling of Malware Propagation, Technical Report TR-08-017, Department of Computer Sciences, Purdue University, 2008

[14] A.G. Voyiatzis and D.N. Serpanos, Pulse: A Class of Super-Worms against Network Infrastructure. Proc. of ICDCS Workshops, May 2003

[15] A. Wagner, T. Dubendorfer, B. Plattner, and R. Hiestand, Experiences with Worm Propagation Simulations, Proc. of ACM Workshop on Rapid Malcode, Oct. 2003

[16] NMAP documentation, http://nmap.org/book/man-performance.html, last accessed *Jul. 1, 2009*

[17] Kademlia Specification, http://xlattice.sourceforge.net/components/protocol/kademlia/specs.html, last accessed *Jul. 1, 2009*

[18] Z. Chen and C. Ji, A Self-Learning Worm Using Importance Scanning, ACM Workshop on Rapid Malcode, Nov. 2005

[19] S. Friedl, Analysis of the new Code Red II Variant, http://www.unixwiz.net/techtips/CodeRedII.html, Last accessed *Apr. 3, 2009*

[20] T. Koshy, Fibonacci and Lucas Numbers with Applications, Wiley-Interscience, Aug. 2001

[21] Editorial, A Cyberblockade in Estonia, New York Times, Jun. 2, 2007

[22] I. Dumitriu, On generalized Tribonacci sequences and additive partitions, Discrete Mathematics, Vol. 219 , Iss. 1-3, 2000

[23] D. Dagon, G. Gu, C. Lee, and W. Lee. "A Taxonomy of Botnet Structures.", Proc. of the 23rd Annual Computer Security Applications Conference (ACSAC), Dec. 2007.