The Effect of DNS Delays on Worm Propagation in an IPv6 Internet

Abhinav Kamra, Hanhua Feng, Vishal Misra and Angelos D. Keromytis
Department of Computer Science
Columbia University in the City of New York
Email: {*kamra,hanhua,misra,angelos*}*@cs.columbia.edu*

***Abstract*— It is a commonly held belief that IPv6 provides greater security against random-scanning worms by virtue of a very sparse address space. We show that an intelligent worm can exploit the directory and naming services necessary for the functioning of any network, and we model the behavior of such a worm in this paper. We explore via analysis and simulation the spread of such worms in an IPv6 Internet. Our results indicate that such a worm can exhibit propagation speeds comparable to an IPv4 random-scanning worm. We develop a detailed analytical model that reveals the relationship between network parameters and the spreading rate of the worm in an IPv6 world. We also develop a simulator based on our analytical model. Simulation results based on parameters chosen from real measurements and the current Internet indicate that an intelligent worm can spread surprising fast in an IPv6 world by using simple strategies. The performance of the worm depends heavily on these strategies, which in turn depend on how secure the directory and naming services of a network are. As a result, additional work is needed in developing detection and defense mechanisms against future worms, and our work identifies directory and naming services as the natural place to do it.**

Keywords: Stochastic Processes/Queuing Theory, Simulations, Worm Propagation.

I. Introduction

In recent years, the Internet has been plagued by a number of worms [1], [2], [3], [4]. Many of these worms use random address scanning to identify new hosts to infect. The Slammer [5] and Witty [6] worms amply demonstrated the effectiveness of this brute force technique in spreading at time scales that do not permit human reaction and make automated reaction very difficult. Arguably, the effectiveness of random scanning owes to the fact that IPv4 addresses are only 32 bits long, thus allowing for an fast exhaustive search of all possible hosts, and the relative population (host) density in this space.

Following this reasoning, it is natural to expect that the eventual adoption of IPv6 [7] will affect the propagation speed of scanning worms. In particular, the 128-bit IPv6 addresses should make it considerably more difficult for a worm to find new targets through random selection. Assuming that the total number of hosts on the Internet does not increase by a similar factor, the work factor for finding a target in an IPv6 Internet will increase by approximately 2^{96}, rendering random scanning prohibitively expensive. Note that email worms, which use the address books and other information resident on infected machines to identify new targets, will not be affected by the adoption of IPv6.

However, we believe that future worms are likely to use other, more effective strategies in identifying and targeting likely targets. In particular, we expect worms to use a two-level scanning hierarchy, whereby different mechanisms are used when scanning across subnets and when scanning inside subnets. This approach, which has already been used by some worms [2], [8], exploits the fact that scanning and propagation speeds inside a local network are considerably higher, due to lower latency and higher bandwidth. In an IPv6 network, a second consideration is the ease of locating additional likely targets once one node on the network has been infected. For example, routing protocols, Windows service location announcements, neighbor discovery caches, and host configuration and log files can be used to identify additional hosts on the local network. Thus, the main obstacle faced by a scanning worm in IPv6 is how to locate valid networks, and at least a small number of hosts in those networks.

One strategy we identify is *DNS random scanning, i.e.,* a worm that guesses DNS names instead of IP addresses, and uses the DNS infrastructure to locate likely targets. Although the inherent cost of a DNS query can be significantly larger than that of a simple probe at a random addresses, we find through simulations that a worm that pipelines DNS queries and attempts infections asynchronously (*i.e.,* as DNS replies are received) can exhibit propagation speeds very close to those of random-scanning worms in the current IPv4 Internet. This is a particularly surprising and worrisome finding, as it greatly diminishes the prospect of inherently better security in an IPv6 Internet. After our submission of this paper, one such worm appeared in the IPv4 Internet, named MyDoom [9].

Our analytical models indicate that the spread of such DNS-based worms is greatly influenced by the variance of the fractions of hosts currently infected in various subnetworks. The higher the variance, the slower is the growth. Thus, a fast-spreading worm should spread out as uniformly as possible across different subnets, and hence it needs an effective strategy for identifying vulnerable hosts in different subnets. Our results also demonstrate that the speed of spread depends to a great extent on the strategy employed in locating additional

This material was supported in part by the National Science Foundation under Grant No. CAREER ANI-0238299 and Grant No. CCR-TC-0208972, and by gifts from the Intel IT Research Council, CISCO URP, and IBM. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

targets within a network once a host has been infected. Hence, security should be tightened against strategies that allow easy or unauthorized access to valid IPv6 addresses in a local network.

Thus, we believe that further research is needed in developing mechanisms for detecting and responding to fast-spreading worms. One natural initial counter-measure to the "DNS worm" is to install anomaly detection capabilities close to DNS servers. This will help in identifying likely worm infestations by measuring the rate at which hosts generate DNS queries, although it is unlikely to eliminate the worm problem by itself. While the scenario that we study in this paper concerns DNS and IP addresses, the general principles apply to any situation where "targets" are identified by employing a directory or search service. We hope that our work will incentivize additional work in the area of worm detection and countermeasures.

Paper Organization: The remainder of this paper is organized as follows. Section II gives a brief background on the DNS infrastructure, along with a simple analytical model for the cost of queries. Section III discusses the DNS worm, and models its propagation speed. Section IV briefly describes the simulator we use in Section V, which contains our results on projected worm propagation using both mathematical models and the DNS worm simulator. Section VI discusses related work on worm detection and defenses. We conclude the paper with Section VII.

II. Background on DNS

DNS provides a mapping from alphabetical domain names to the numerical IP addresses used to identify hosts in the Internet. The DNS architecture is a hierarchy of distributed "name-servers" that contain databases of name-to-IP mappings. In a typical DNS query, a client needs to obtain the IP address for a distant host it needs to contact. It first contacts the local "resolver", a DNS server in the same domain as the client. This resolver then contacts one of the root name-servers that are at the top of the DNS hierarchy. The resolver is then recursively referred to a succession of name-servers down the hierarchy until it queries the *authoritative name-server* for the hostname to be resolved. The *authoritative name-server* then replies to the local resolver with the required IP address. The local resolver then sends it to the client and also caches a copy for immediate retrieval in case of further queries for the same hostname from a client in the domain for which it is the local resolver. The logical path taken by a typical DNS query is shown in Figure 1.

The time taken for a DNS query consists of round-trip delays between the local resolver and the client and also the round-trip delays between the local resolver and the name-servers queried. In mathematical form,

$$d = d_{local} + d_{internet} \tag{1}$$

where d_{local} is the round-trip delay between the client and the resolver and $d_{internet}$ is the elapsed time between the time the resolver issues the DNS query to the root name-server and the time it gets the response back from the authoritative name-server. The delay $d_{internet}$ may consist of round-trip times of communication amongst multiple pairs of hosts. These round-trip delays in turn depend on a multitude of factors:

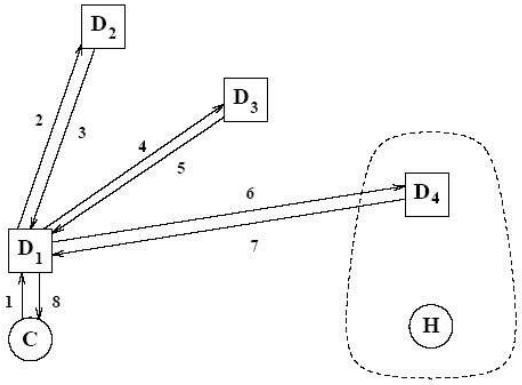

Fig. 1. A typical DNS Query path: C is the client. D_1 is the local DNS resolver, D_2, D_3, D_4 are DNS servers with D_4 the authoritative DNS server for host H.

1) **Timeouts and Retransmissions***:* Packet losses due to congestion in the network may trigger retransmissions that increase the DNS delay. If a DNS query packet is lost, typically the client waits for a timeout T before sending a retransmission. If p_r is the loss probability (and hence the retransmission probability), then the expected DNS delay for a query is

$$d_{av} = d_{local} + d_{internet} + \frac{p_r T}{1 - p_r} \tag{2}$$

We assume that the local resolver resides in the same subnetwork as the client and so the delay d_{local} is not much affected by the load on the DNS servers.

2) **DNS Cache Hit/Miss***:* The hostname to be resolved may already be present in the cache of the local resolver, in which case the DNS delay is considerably less. In essence, from the client's perspective, the DNS delay depends on *(i)* the Cache Hit/Miss probability and *(ii)* Congestion in the Internet (which may affect the round-trip delays).

If p_{DCH} is the probability of a DNS cache hit when resolving a hostname, then the average DNS delay may be written as

$$\begin{aligned} d_{av}^{cached} &= p_{DCH} * d_{local} + (1 - p_{DCH}) * d_{av} \\ &= d_{local} + (1 - p_{DCH}) \\ &\cdot \quad (d_{internet} + \frac{p_r T}{1 - p_r}) \end{aligned} \tag{3}$$

III. Worm Propagation Model

A. Random Scanning

In a typical random scanning worm (such as versions of CodeRed and Slammer worm) propagating in an IPv4 Internet

space, each worm instance generates random IP addresses and tries to infect the host with that IP address. Various earlier works [10], [11] have modeled this random scanning worm using a simple epidemic model. The assumption is that any Internet host is either vulnerable to infection or has already been infected, in which case it contributes to the worm propagation by infecting other machines. Also, once a host has been infected, it remains infected. The classical simple epidemic model is given by the equation:

$$\frac{dI(t)}{dt} = \beta I(t)[N - I(t)] \tag{4}$$

where β is the pairwise rate of infection [12]. At $t = 0$, $I(0)$ hosts are infected and start the process of infecting the remaining $N - I(0)$ hosts.

β is given by $\beta = \frac{\xi}{\Omega}$ where ξ is the scan rate of the worm and Ω is the scanning space. For instance, in the case of the Slammer worm [5], each infected host sent out 4000 scans per second and hence $\xi = 4000/s$. Since the Slammer worm generates random IP addresses from the whole IPv4 space consisting of 2^{32} IP addresses, we have $\Omega = 2^{32}$.

In an IPv6 Internet, random scanning worms run into insurmountable problems since the scanning space Ω is huge for IPv6. This results in extremely low values of the pairwise infection rate β and hence very slow propagation for worms that propagate at reasonable speeds in IPv4.

The fraction of infected hosts at any given time t is denoted by $a(t) = \frac{I(t)}{N}$. The dynamics of a follow an equation similar to Equation (4).

B. The DNS Worm

The DNS Worm overcomes this obstacle by not relying on random scanning. The DNS worm uses DNS queries to find active IP addresses in the sparse IPv6 address space. It consists of two parts. At the back-end is an address generator that generates strings which might be actual hostnames on the Internet. The front-end then uses DNS resolution to find the corresponding IP address, which is then attacked and infected, if deemed vulnerable.

1) String Generation: The DNS Worm back-end consists of a string generator that generates strings which are probable names of actual hosts on the Internet. Internet hostnames are typically made up of common words separated by dots (*e.g.,* www.yahoo.com). Most of the words used are dictionary words; some prefixes and suffixes such as "www" and ".com" respectively, even though not dictionary words, are extremely common. Thus, a smart DNS Worm can use a form of dictionary attack to generate probable Internet hostnames. Apart from dictionary-based string generation, a worm can use web search engines to gather valid hostnames, and in particular server names. Still more hostnames that are not necessarily web servers and hence do not show up on web search engines, can be retrieved from other public access Internet locations such as Google groups, mailing lists, *etc*. Recently, a variant of an email worm known as *MyDoom*, harvested email addresses by sending search queries to popular web search engines as it spread [9]. This slowed the search engines considerably, in some cases totally knocking them out [†]. Another worm called *Santy* used the search engine Google to find websites containing online bulletin boards running a vulnerable version of the widely used PHP Bulletin Board(phpBB) software. By using similar sophisticated techniques, the string generator can produce actual host addresses with high probability.

We denote the set of all possible strings which can be produced by the string generator as χ. The subset of χ that are actual host addresses is denoted by χ^{target}. An instance of the DNS Worm that uses the string generator to produce probable host addresses and then tries to infect the valid addresses is only able to infect hosts from the set χ^{target}. Naturally, there are still valid Internet host addresses that lie outside χ but which cannot be produced by the string generator and as a consequence cannot be infected. Thus, from the view of the DNS Worm, the vulnerable hosts on the Internet are only the hosts with addresses contained in string set χ^{target}. Hence for all analytical purposes, $N = \chi^{target}$.

The DNS Worm operates by iterating over two steps:

- Generate a new probable hostname using the string generator.
- Resolve the probable hostname by initiating a DNS query. If a valid IP address is returned, it implies that there is an actual Internet host with this name. In this case the host is attacked and infected. The DNS query may also result in no corresponding IP address being found.

For a string produced by the string generator, the probability of it being a valid hostname is

$$\sigma = \frac{\chi^{target}}{\chi} \tag{5}$$

Note that we assume that all these χ^{target} hosts have the vulnerability which the DNS Worm exploits. In case only some fraction of the χ^{target} hosts have the vulnerability, the parameter σ can be scaled accordingly.

2) Effective Scan Rate: For each scan that the DNS Worm performs, it has to perform a DNS query and, if the query is successful, infect the resulting IP address (if vulnerable). The total time taken in the process is the sum of the DNS delay and the infection time. On the other hand if the DNS query is unsuccessful, the worm immediately starts generating a new string for the next probable infection. Hence the total time is just the DNS delay. Since the DNS query was unsuccessful, the string was not a valid hostname and hence cannot be found in the DNS Domain-Local Cache. Therefore, the average delay for such queries is $d_{av}(a)$. The average delay for successful queries is $d_{av}^{cached}(a) + \tau_f$ where τ_f is the average infection time.

Note that these delays are a function of a, the fraction of infected hosts (as defined in Section III-A), since, as the number of infected hosts goes up, so does the DNS traffic due to the worms and hence the load on the DNS servers, which in turn affects the DNS delays.

[†]Note that this occurred after the submission of this paper to Infocom 2005.

As observed in Equation (5), the probability of querying for valid hostnames (and hence of successful DNS queries) is given by σ. The effective average DNS delay for a worm then becomes

$$d_{eff}(a) = \sigma(\tau_f + d_{av}^{cached}(a)) + (1-\sigma)(d_{av}(a)) \quad (6)$$

Hence the effective scan rate of the DNS worm is given by

$$\xi(a) = \frac{1}{\sigma(\tau_f + d_{av}^{cached}(a)) + (1-\sigma)(d_{av}(a))} \quad (7)$$

3) DNS Worm Propagation Rate: We now derive the dynamics of a, the fraction of infected machines. Recall that a is the fraction of "vulnerable" machines that have been infected. If $I(t)$ is the number of machines infected at time t, then (since from the point of view of the DNS Worm the number of vulnerable machines is χ^{target}) we have $a(t) = \frac{I(t)}{\chi^{target}}$.

Consider an infinitesimal time period δ. If the average scan rate of infected machines is ξ, then in time δ, $I(t)$ machines can perform $\xi\delta I(t)$ number of scans. Out of these, since $\sigma = \frac{\chi^{target}}{\chi}$ is the probability of a DNS Worm producing a string which is a valid Internet host, the number of scans (and so the number of DNS queries) that return a valid IP address are $\sigma\xi\delta I(t)$. Since a fraction of the vulnerable hosts are already infected, the probability of an IP address retrieved using a DNS query belonging to a still uninfected host is $1 - a(t)$.

Hence the new infections in time period δ are

$$I(t+\delta) - I(t) = \sigma\xi\delta I(t) * (1 - \frac{I(t)}{\chi^{target}})$$
$$\Rightarrow \frac{dI(t)}{d(t)} = \sigma\xi I(t) * (1 - \frac{I(t)}{\chi^{target}})$$

Since $a(t) = \frac{I(t)}{\chi^{target}}$, we have

$$\dot{a} = \sigma\xi a(1-a)$$

III-B.2 shows how ξ also varies with a. Thus, we have the differential equation governing the dynamics of a given by:

$$\dot{a} = \frac{\sigma a(1-a)}{\sigma(\tau_f + d_{av}^{cached}(a)) + (1-\sigma)(d_{av}(a))} \quad (8)$$

C. Modeling the DNS architecture and the Resulting Delays

The DNS architecture consists of a hierarchy of DNS servers that respond to DNS queries. At the top of the hierarchy are the Root DNS servers. Most of the DNS queries that are not locally resolved are at first directed to one of these Root DNS servers. During periods of fast propagation of the DNS Worm, the number of DNS queries increases dramatically, possibly overloading the Root DNS servers, which then become the bottleneck. The Root DNS servers have a bounded processing power. To study the bounded throughput behavior of Root DNS servers, we model them as an M/M/1/K queuing system. This is just a first order approximation and in no way implies that the actual Root server behavior follows the M/M/1/K queuing model.

The queuing system serves DNS queries and has an exponential service rate given by μ. K is the maximum number of queries that can be present (either waiting or being served) in the queuing system at a given time. During times of high load, not all queries can be served. Many of the queries will be dropped due to buffer exhaustion in the queuing system.

If λ is the arrival rate of queries, then the probability of the queue having i queries waiting to be served is given by

$$\pi(i) = \frac{(1-\rho)\rho^i}{1-\rho^{K+1}} \quad (9)$$

where ρ is the load on the system and is given by $\rho = \frac{\lambda}{\mu}$.

The expected probability of a query being dropped due to buffer exhaustion is given by

$$E[loss] = \pi(K) = \frac{(1-\rho)\rho^K}{1-\rho^{K+1}} \quad (10)$$

Queries are accepted in the system when the M/M/1/K queue is not full. The mean expected response time of only the accepted queries is then given by

$$\begin{aligned} E[X_a] &= E[X|accepted] = \frac{E[X]}{1-E[loss]} \\ &= \frac{1}{1-\pi(K)}\frac{1}{\mu}\sum_{i=0}^{K-1}(i+1)\pi(i) \\ &= \frac{1}{\mu}[\frac{1}{1-\rho} - \frac{K\rho^K}{1-\rho^K}] \end{aligned} \quad (11)$$

For the special case of $\rho = 1$, we have

$$\begin{aligned} \pi(i)_{\rho=1} &= Lim_{\rho\to 1}\frac{(1-\rho)\rho^i}{1-\rho^{K+1}} = \frac{1}{K+1} \\ E[loss]_{\rho=1} &= \pi(K) = \frac{1}{K+1} \\ E[X_a]_{\rho=1} &= \frac{E[X]_{\rho=1}}{1-E[loss]_{\rho=1}} = \frac{K+1}{2\mu} \end{aligned} \quad (12)$$

The query arrival rate λ depends on how many hosts are sending DNS queries. The higher the number of infected hosts, the more DNS queries will be received by the name-servers. Hence λ will increase with a, the fraction of infected hosts, which in turn implies that $E[X_a]$ and $E[loss]$ are functions of a. If ξ is the scan rate of infected hosts, then $\lambda(a)$ will be the number of infected hosts times the scan rate. That is $\lambda = aN\xi$. Thus, we have

$$\rho(a) = \frac{aN\xi}{\mu} \quad (13)$$

D. DNS Worm Propagation Revisited

From Equation (8), we see that the rate of worm propagation depends on DNS delay values $d_{av}(a)$ and $d_{av}^{cached}(a)$ that are given by Equations (2) and (3).

The system of Root DNS servers is modeled as an M/M/1/K queuing system in section III-C. The expected response time of the queuing system for accepted queries is thus a good measure of the average time spent by a typical DNS query in the Internet. Using Equation (11) we have $d_{internet}(a) = E[X_a]$.

The expected probability of a DNS query being dropped in the M/M/1/K queuing model is a good measure of the retransmission probability of a DNS query by the worm. Using

Equation (10), we have $p_r(a) = E[loss]$. Note that $d_{internet}$ and p_r are functions of a, since $E[X_a]$ and $E[loss]$ are also functions of a.

Using Equations (2) and (3), we have

$$\begin{aligned} d_{av}(a) &= d_{local} + d_{internet} + \frac{p_r T}{1-p_r} \\ &= d_{local} + E[X_a] + \frac{p_r T}{1-p_r} \\ &= d_{local} + \frac{p_r T}{1-p_r} \\ &+ \frac{1}{\mu}[\frac{1}{1-\rho} - \frac{K\rho^K}{1-\rho^K}] \end{aligned} \tag{14}$$

$$\begin{aligned} d_{av}^{cached}(a) &= d_{local} + (1-p_{DCH})(d_{internet} + \frac{p_r T}{1-p_r}) \\ &= d_{local} + (1-p_{DCH})(E[X_a] + \frac{p_r T}{1-p_r}) \\ &= d_{local} + (1-p_{DCH})(\frac{p_r T}{1-p_r} \\ &+ \frac{1}{\mu}[\frac{1}{1-\rho} - \frac{K\rho^K}{1-\rho^K}]) \end{aligned} \tag{15}$$

where $\rho = \frac{aN\xi}{\mu}$.

Finally, we have the rate of propagation of the DNS Worm given by

$$\dot{a} = \frac{\sigma a(1-a)}{\sigma(\tau_f + d_{av}^{cached}(a)) + (1-\sigma)(d_{av}(a))} \tag{16}$$

where $d_{av}(a)$ and $d_{av}^{cached}(a)$ are given by Equations (14) and (15).

E. The Two-level Model

The epidemic model of a uniform scanning worm described in III-A does not capture the behavior of many existing worms that differentiate the IP addresses of the same IPv4 subnet to arbitrary IP addresses (*e.g.*, CodeRed2). This locality property becomes much more important in IPv6 networks. The local IPv6 address space is already too large for a worm to perform random scans just by guessing IP addresses. However, there are many more efficient ways to find a host in the local network. Routing protocols, Windows service location announcements, neighbor discovery caches, and host configuration and log files can be exploited to identify additional hosts on the local network.

In the previous sections, we proposed to use name-servers to search for hosts in an IPv6 Internet, which is much less efficient than possible methods that explore the local network. An effective IPv6 worm has to consider the locality of the Internet and use different propagation methods: a global scan method and a local scan method. The global scan method is inefficient but necessary, because it can cover a large portion of the total population of the vulnerable hosts on the Internet. The local scan method is efficient but can only discover vulnerable hosts on the local network. This results in a much higher infection rate to hosts in the local network than against arbitrary host on the Internet. As a result, we use a two-level model to describe the propagation of an IPv6 worm.

Suppose the population of vulnerable hosts in local network i is N_i. For an infected host in this local network, let Ω_i be the search space of addresses of local scans. Quantity Ω_i is not necessarily very large; it can be just the population of all (vulnerable and non-vulnerable) hosts on the local network. Suppose there are n such local networks on the Internet, and the total vulnerable hosts on the Internet is $N = \sum_{i=1}^{n} N_i$. Let Ω be the search space for global scans, that is actually the cardinality of the set of names that we feed to name-servers. Clearly, we can assume a very large Ω, much greater than $\sum_{i=1}^{n} \Omega_i$, although it may be much less than 2^{128}, the cardinality of the IPv6 address space. We denote by ξ_i and ξ the local and global scan rates. For a worm using pure random scanning, ξ_i is greater than ξ due to shorter round-trip times and greater available bandwidth within a local network. With a DNS Worm, ξ_i can be much greater than ξ due to DNS delays. The rates at which an individual host in local network i is locally and globally scanned are $(\xi_i/\Omega_i)I_i$ and $(\xi/\Omega)I$ respectively. Therefore, the infection rate of a new host in local network i is

$$R_i = \left(\frac{\xi_i I_i}{\Omega_i} + \frac{\xi I}{\Omega}\right)(N_i - I_i). \tag{17}$$

It is not surprising that the locality considerably affects worm propagation – in fact, we can analyze it with a continuous model derived from (17), which is

$$\frac{dI_i}{dt} = \left(\frac{\xi_i I_i}{\Omega_i} + \frac{\xi I}{\Omega}\right)(N_i - I_i). \tag{18}$$

Let $A_i = I_i/N_i$ be the fraction of infected hosts, and also assume all local networks are homogeneous, *i.e*, all N_i's, ξ_i's and Ω's are respectively identical and $N_i = N/n$. Then, we obtain

$$\frac{dA_i}{dt} = \left(\frac{\xi_i N_i}{\Omega_i} A_i + \frac{\xi}{\Omega}\sum_{j=0}^{n} A_j N_j\right)(1 - A_i) \tag{19}$$

$$= \left(\frac{\xi_i N_i}{\Omega_i} A_i + \frac{\xi N}{\Omega}\left(\frac{1}{n}\sum_{j=0}^{n} A_j\right)\right)(1 - A_i). \tag{20}$$

Summing on both sides over $i = 1, \ldots, n$, and observing that the global fraction of infected host $a = \frac{1}{n}(\sum_{i=1}^{n} A_i)$, we obtain

$$\begin{aligned} &\frac{da}{dt} \\ &= \frac{1}{n}\sum_{i=0}^{n}\left[\left(\frac{\xi_i N_i}{\Omega_i} A_i + \frac{\xi N}{\Omega}\left(\frac{1}{n}\sum_{j=0}^{n} A_j\right)\right)(1 - A_i)\right] \\ &= \frac{\xi_i N_i}{\Omega_i}\left(\frac{1}{n}\sum_{i=0}^{n} A_i\right) + \frac{\xi N}{\Omega}\left(\frac{1}{n}\sum_{j=0}^{n} A_j\right) \\ &\quad - \frac{\xi_i N_i}{\Omega_i}\left(\frac{1}{n}\sum_{i=0}^{n} A_i^2\right) - \frac{\xi N}{\Omega}\left(\frac{1}{n}\sum_{j=0}^{n} A_j\right)^2 \end{aligned}$$

$$= \left(\frac{\xi_i N_i}{\Omega_i} + \frac{\xi N}{\Omega}\right) a - \frac{\xi N}{\Omega} a^2 - \frac{\xi_i N_i}{\Omega_i} \left(\frac{1}{n}\sum_{i=0}^{n} A_i^2\right)$$

$$= \left(\frac{\xi_i N_i}{\Omega_i} + \frac{\xi N}{\Omega}\right) a - \left(\frac{\xi N}{\Omega} + \frac{\xi_i N_i}{\Omega_i}\right) a^2 - \frac{\xi_i N_i}{\Omega_i} \left(\frac{1}{n}\sum_{i=0}^{n} (A_i - a)^2\right)$$

$$= \beta a(1-a) - \frac{\xi_i N_i}{\Omega_i} \left(\frac{1}{n}\sum_{i=0}^{n} (A_i - a)^2\right), \quad (21)$$

where $\beta = \left(\frac{\xi_i N_i}{\Omega_i} + \frac{\xi N}{\Omega}\right)$ is the sum of infection rates of global and local scanning, and it is identical for all i's with our assumption. Note that the last item in (21) is the product of the local infection rate $\xi_i N_i/\Omega_i$ and the variance of A_i's, which is non-negative. If all subnets have identical numbers of initial infected hosts, this item is zero and the model is simplified to the simple epidemic model. With a variation of fractions of infected hosts in local networks, the global average infection rate will decrease by a rate proportional to the variance of fractions.

IV. The DNS Worm Simulator

We use a simulator to analyze the propagation of IPv6 worms with the models in Section III. There may be thousands to millions of vulnerable hosts on the Internet, so it is impossible to simulate them individually. Even if we do not consider computational complexity, it is hard to identify representative configurations. For this reason, we simulate each local network as a group. In our simulation, we consider the scan of each infected host as a stochastic process. The time between two scans is random and may satisfy certain distributions. We assume the scanning processes of different hosts are fairly independent. For worms using multi-threading, we consider each thread as an independent stochastic process. The probabilities that global and local scans from an infected host reach a certain vulnerable host are $1/\Omega_i$ and $1/\Omega$, respectively, and are both very small. Then, regardless of the nature of the infection mechanism, the stochastic process for the number of a local network is close to a Poisson process with rate R_i, due to many Bernoulli selections with small probabilities and the summation of independent processes. With this assumption, we simulate the worm propagation by dividing the whole Internet into n counting processes that represent n local networks. Each counting process is a Poisson process with a changing rate, which is R_i for the i-th local network. We assume there is an initial population of infected hosts and denote it by $I^0 = \sum_{i=0}^{n} I_i^0$.

V. Simulation Experiments

In this section, we study the propagation rates of various kinds of DNS worms based on our model in the earlier sections and the effect of various parameters.

Although the address space of IPv6 is 2^{96} times greater than that of IPv4, the total number of hosts on an IPv6 Internet is expected to be only a few orders of magnitude greater than what it is now.

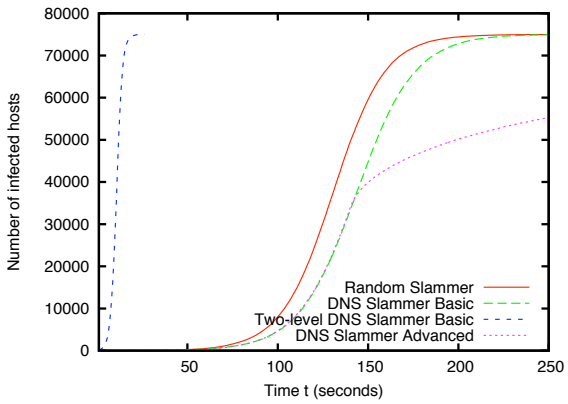

Fig. 2. Comparison of various versions of DNS-Slammer worms *vs* the original Slammer worm propagating in IPv4

Table I shows the various parameters related to two different well-studied worms, Slammer and CodeRed. We use DNS Worm parameters comparable to Slammer and CodeRed parameters in evaluating the propagation.

We define two different types of DNS Worms. The first, referred to as DNS-Basic Worm, incurs only constant DNS delays for all its DNS queries. The other version, referred to as DNS-Advanced Worm, incurs DNS delays based on the DNS delay model described in III-C. The Simulator uses Equation (16) to simulate the propagation of the DNS-Advanced Worm.

The DNS Worm also has parameters such as the maximum number of vulnerable machines N, which are common with earlier IPv4 worms such as Slammer and CodeRed. We refer to DNS-Slammer as the worm that has all such common parameters the same as the Slammer worm. Similarly, the DNS-CodeRed worm has all the common parameters the same as the CodeRed worm. This is further shown in Table I.

We can have combinations of DNS Worm characteristics. For example, DNS-Basic-Slammer is the DNS Worm with Slammer parameters and constant DNS delays.

For the simulation of the DNS-Advanced Worm, we also need the values of the DNS model parameters μ and K. We choose them to be $\mu = 5 \times 10^4/second$ and $K = 1000$.

A. Comparison with IPv4 Worms

We now examine how the DNS Worm propagates in the IPv6 Internet space, compared to earlier worms such as Slammer and CodeRed in the IPv4 Internet space.

Figure 2 shows how the DNS Worm propagation in IPv6 compares with that of the Slammer worm in an IPv4 environment. The DNS-Slammer-Basic worm (DNS Worm with Slammer parameters and constant DNS delays) is able to propagate almost as fast as the Slammer worm. The two-level DNS-Slammer-Basic worm has an additional local-subnet propagation rate that makes it extremely fast. Thus, it is able to infect all the vulnerable hosts in as few as 20 seconds. The DNS-Slammer-Advanced worm does slow itself down due to

TABLE I

PARAMETERS TABLE: VARIOUS PARAMETERS USED BY THE WORMS AND CORRESPONDING VALUES FOR SLAMMER AND CODERED WORMS AS IDENTIFIED IN STUDIES (FOR SUBSECTIONS V-A AND V-B ONLY)

Parameter	Description	Slammer Value	CodeRed Value	DNS-Slammer Value	DNS-CodeRed Value
ξ_i	Scan Rate	4000 / second	358 / minute	4000 / second	358 / minute
N	Vulnerable hosts	75000	360000	75000	360000
n	Subnetworks	N/A	N/A	7500	36000
N_i	Hosts in each subnetwork	N/A	N/A	10	10
σ	Probability of successful scans	$75000/2^{32}$	$360000/2^{32}$	1/50	1/50
I^0	Initially infected hosts	10	10	10	10

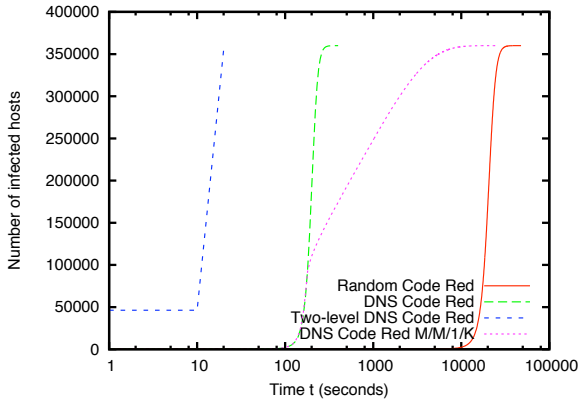

Fig. 3. Comparison of various kinds of DNS-CodeRed worms with the Original CodeRed worm propagating in IPv4

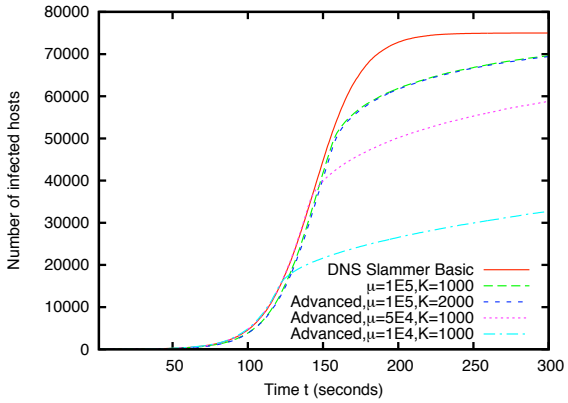

Fig. 4. Effect of varying throughput μ and buffer size K on the various forms of the DNS-Slammer Worm

increasing DNS delays as the infection increases, but is able to propagate comparably with the Slammer worm during the initial stages of infection.

Figure 3 shows results for the same experiments using CodeRed worm parameters for all the worm models. It is interesting to note that the DNS-CodeRed-Basic worm now propagates much faster than the CodeRed worm. This is because CodeRed worm has a much smaller σ (probability of successful scan) than DNS-CodeRed-Basic worm. The DNS-CodeRed-Advanced worm still propagates much faster than the CodeRed worm. Note that the x-axis in Figure 3 is in log-scale, since the two-level worm is much faster than other worms.

B. Effect of Maximum Throughput

In our DNS Worm model explained in III, we observe that the worm propagation rate given by Equation (16) depends on the DNS architecture parameters μ and K, which are respectively the maximum throughput of DNS queries that the DNS architecture can handle and the maximum backlog for the queries that the system can hold at a given time.

In this simulation, we explore the effect μ and K have on the propagation rate of the DNS Worm. For this purpose, we choose various values of μ and K and simulate the propagation of the DNS-Advanced worm. Figure 4 shows how μ and K affect the propagation rate of the DNS-Advanced worm. Figure 5 shows the same simulations done with CodeRed parameters

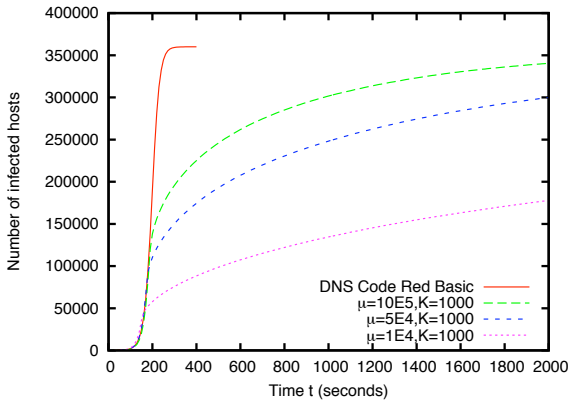

Fig. 5. Effect of varying throughput μ on the various forms of the DNS-CodeRed Worm

with similar results and observations.

As we can see, K does not have much of an impact on the propagation for worm models with the same μ value. On the other hand, increased throughput μ helps the worm to propagate faster. For comparison purposes, we also show the propagation of a DNS-Basic worm that has constant DNS delays and hence does not depend on μ and K values. The interesting thing to note is that the shape of the propagation curves for DNS-Advanced worm models is very different from the DNS-Basic worms. We observe a break-off point where the worm propagation suddenly slows down. This is due to

TABLE II

PARAMETERS TABLE: VARIOUS PARAMETERS USED BY THE ONE-LEVEL AND TWO-LEVEL WORMS (FOR SUBSECTIONS V-C AND V-D ONLY)

Parameter	Description	Value
ξ	Global Scan Rate	0.5 / second
ξ_i	Local Scan Rate	1 / second
N	Vulnerable hosts	10^8
n	Subnetworks	10^4
N_i	Hosts in each subnetwork	10^4
σ	Probability of successful scans	1/50
I^0	Initially infected hosts	1000

the saturation of the queue for the M/M/1/K queuing system described in III-C; furthermore, the high number of queries generated by the spreading worm acts as a negative feedback, self-regulating the spread. This points to a possible defense mechanism, limiting the throughput of the DNS servers to reach the break-off point as early as possible. On the other hand, it is also likely to result in poor performance of DNS lookups for legitimate users. One possible answer is better anomaly detectors at DNS servers. Deploying them only at the root DNS servers may be sufficient, depending on their accuracy.

C. Effect of Initial Variance on Propagation Rate

Section III-E shows that the two-level worm propagation rate is affected by the variance of number of infected hosts in different local subnetworks. Here, we examine the effect of initial variance in the distribution of infected hosts in the local subnetworks on the propagation rate of the DNS-Basic Worm.

For this experiment, we set the total number of vulnerable hosts to 10^8, with each local subnetwork having 10^4 vulnerable hosts. The local scan rate ξ_i is supposed to be 1 per second and the global scan rate ξ is 0.5 per second. We assume that the worm can efficiently discover all existing vulnerable hosts on a local subnetwork, which means $\Omega_i = N_i$. For the global scanning, we set $\sigma = N/\Omega = 1/50$. Initially, the worm has already infected $I^0 = 1000$ hosts. These parameters are listed in Table II. These 1000 hosts may be distributed in various ways amongst the 10^4 local subnetworks, resulting in various variance levels. Figure 6 shows the worm propagation as a function of time for different initial variance values. Note that curve *(e)* in Figure 6 is an analytical result for the ideal case of no variance throughout the simulation, although it is impossible for the infected hosts to be uniformly distributed amongst the local subnetworks at all times. As we can see observe from Figure 6, as the variance increases, the worm propagation becomes slower.

Figure 7 shows the correlation of the initial variance and the time for the worm to infect 80% of the vulnerable hosts. We can see that the initial variance of the distribution of the infected hosts in the local subnetworks has a pronounced effect on the infection time.

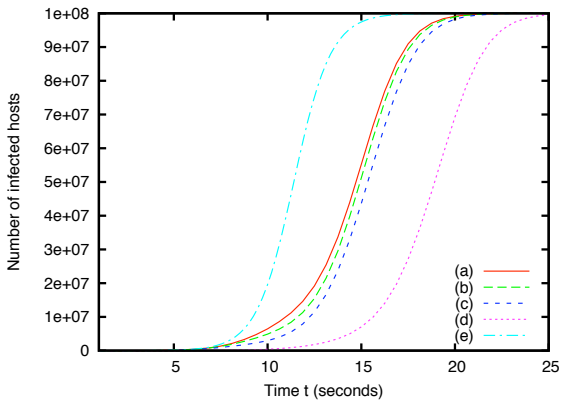

Fig. 6. The numbers of infected hosts as functions of time in seconds, with 1000 infected hosts at time 0. (a) Each of 1000 local networks has one infected host. (b) Each of 500 local networks has two sends. (c) Each of 200 local networks has five infected hosts. (d) Only One local network has all 1000 seeds. (e) the simple (one-level) epidemic model with a rate of the sum of the global and local rates

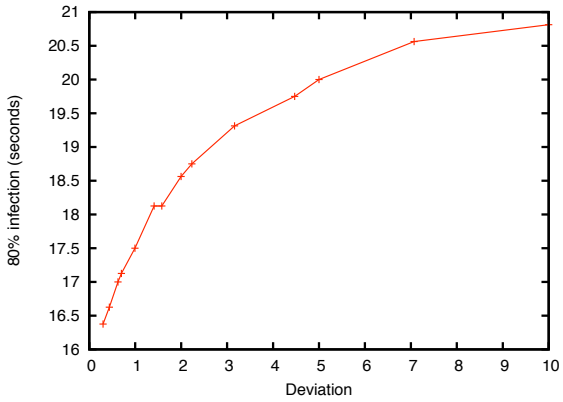

Fig. 7. Time to infect 80% of vulnerable hosts as a function of initial variance of infected host distribution

D. Effect of Local Scanning Rate

The various experiments in the previous subsections show that the Two-level DNS Worm is much faster than one-level worms. This is typically due to much faster propagation rates in the local subnetworks than across networks.

We now examine the effect of local propagation on the overall propagation rate of the Two-level DNS-Basic worm. We use the parameter values from Table II. Figure 8 shows how the local scanning rate affects the total worm propagation. It is important to note that the pronounced effect of decreased local scanning rate is to prolong the initial infection period. Thus, for example, it takes much longer for the worm with smaller local scanning rate to infect 20% of the vulnerable hosts. After that, the infection proceeds pretty smoothly, albeit still slower than the corresponding worm with a higher local scanning rate.

VI. RELATED WORK

Computer viruses are not a new phenomenon, and they have been studied extensively over the last several decades. Cohen

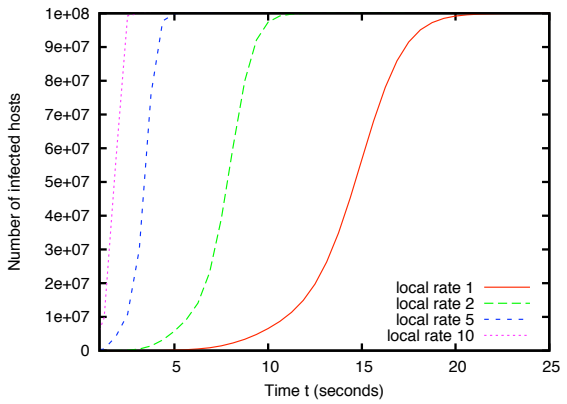

Fig. 8. Effect of local scanning rates on the overall propagation rates of the Two-level DNS-Basic Worm

was the first to define and describe computer viruses in their present form. In [13], he gave a theoretical basis for the spread of computer viruses. In [10], the authors describe the risk to the Internet due to the ability of attackers to quickly gain control of vast numbers of hosts. They argue that controlling a million hosts can have catastrophic results because of the potential to launch distributed denial of service attacks and potential access to sensitive information that is present on those hosts. Their analysis shows how quickly attackers can compromise hosts using "dumb" worms and how "better" worms can spread even faster. The strong analogy between biological and computer viruses led Kephart *et al.* to investigate the propagation of computer viruses based on epidemiological models. In [14], the authors extend the standard epidemiological model by placing it on a directed graph, and use a combination of analysis and simulation to study its behavior. They conclude that if the rate at which defense mechanisms detect and remove viruses is sufficiently high, relative to the rate at which viruses spread, they are adequate for preventing widespread propagation of viruses.

Since the first Internet-wide worm [8], considerable effort has gone into preventing worms from exploiting common software vulnerabilities by using the compiler to inject run-time safety checks into applications (*e.g.,* [15]), safe languages and APIs and static (*e.g.,* [16]) or dynamic [17], [18] analysis tools.

The CodeRed worm [3] was analyzed extensively in [11]. The authors of that work conclude that even though epidemic models can be used to study the behavior of Internet worms, they are not accurate enough because they cannot capture some specific properties of the environment these operate in: the effect of human countermeasures against worm spreading (*i.e.,* cleaning, patching, filtering, disconnecting, *etc.*), and the slowing down of the worm infection rate due to the worm's impact on Internet traffic and infrastructure. They derive a new general Internet worm model called *two-factor worm* model, which they then validate in simulations that match the observed CodeRed data available to them. Their analysis seems to be supported by the data on CodeRed propagation in [19] and [20] (the latter distinguished between different worms that were active simultaneously active). A similar analysis on the SQL "Slammer" (Sapphire) worm [4] can be found in [5]. More recent analyses [21] show that it is possible to predict the overall vulnerable population size using Kalman filters early in the propagation cycle of a worm, allowing for detection of a fast-spreading worm when only 1% or 2% of vulnerable computers on the network have been infected.

CodeRed inspired several countermeasure technologies, such as La Brea [22], which attempts to slow the growth of TCP-based worms by accepting connections and then blocking on them indefinitely, causing the corresponding worm thread to block. Unfortunately, worms can avoid this mechanisms by probing and infecting asynchronously. Under the connection-throttling approach [23], [24], each host restricts the rate at which connections may be initiated. If adopted universally, such an approach would reduce the spreading rate of a worm by up to an order of magnitude, without affecting legitimate communications.

Wong *et al.* [25] study the behavior of the SoBig and MyDoom mass-mailing worms using network packet traces from the CMU network. They identify DNS servers as a possible location for slowing down mass-mailing worms. In constrast, monitoring outgoing mail on SMTP servers is unlikely to work, since most such worms contain their own SMTP engines. Similarly, TCP connection throttling [23], [24] is unlikely to significantly affect mail worm propagation.

[26] describes a design space of worm containment systems using three parameters: reaction time, containment strategy, and deployment scenario. The authors use a combination of analytic modeling and simulation to describe how each of these design factors impacts the dynamics of a worm epidemic. Their analysis suggests that there are significant gaps in containment defense mechanisms that can be employed, and that considerable more research (and better coordination between ISPs and other entities) is needed.

[27] presents some very encouraging results for slowing down the spread of viruses. The authors simulated the propagation of virus infections through certain types of networks, coupled with partial immunization. Their findings show that even with low levels of immunization, the infection slows down significantly.

In the realm of "traditional" computer viruses, most of the existing anti-virus techniques use a simple signature scanning approach to locate threats. As new viruses are created, so do virus signatures. Smarter virus writers use more creative techniques (*e.g.,* polymorphic viruses) to avoid detection. In response detection mechanisms become ever more elaborate, *e.g.,* using partial simulation during program execution. This has led to co-evolution [28], an ever-escalating arms race between virus writers and anti-virus developers.

Lin, Ricciardi, and Marzullo study how computer worms affect the availability of services. In [29], they study the fault tolerance of multicast protocols under self-propagating virus attacks.

VII. Conclusions

In this paper we explored via analysis and simulation the spread of worms in an IPv6 Internet. We modeled and analyzed an *intelligent* worm, that exploits DNS as a means of identifying potentially vulnerable IP(v6) addresses, and uses a two-level spreading mechanism to infect other hosts. Our results demonstrate that by using simple strategies to identify hosts across the two levels, a DNS-based worm in IPv6 can spread as fast as a random-scanning worm in an IPv4 world. This goes against the commonly held belief that IPv6 provides inherently higher security through its larger address space. Our model also identifies the directory and naming service in a network as a potential launching pad for worm attacks in a network with a sparsely populated address space. We explored two scenarios, one in which DNS delays are constant, and another in which DNS delays grow as a function of the number of infected hosts. Experiments with the latter scenario indicate that the spread of the worm is influenced by the processing capacity of the DNS servers. If the spread of the worm results in query volumes that can overwhelm the DNS servers, this can cause DNS service unavailability for legitimate users and induce a denial of service effect. Thus, to protect future networks, DNS (or any directory) servers are a natural location to install anomaly detection and defense capabilities.

Another finding from our analytical model is that the variance of the fractions of hosts infected in a subnet has a big impact on the spreading rate. The more spread-out the worm starts across different subnets, the faster it can infect all vulnerable hosts. We also assume the worm has an efficient way to identify valid IP addresses in a local network, by using techniques like accessing routing protocols, Windows service location announcements, neighbor discovery caches, and host configuration and log files. Our results show that if the second-layer identification mechanism can be hindered, that further slows down the spread of the two-level DNS worm. In summary, directory and naming services, which are critical to the functioning of any network, can also be exploited by intelligent worms to infect hosts. Thus, future IPv6 networks need to shore up the security of the naming and directory services to prevent the spread of such worms.

References

[1] J. F. Reynolds, "Helminthiasis of the Internet," RFC 1135, Internet Engineering Task Force, Dec. 1989.

[2] "CERT Advisory CA-2001-26: Nimda Worm," `http://www.cert.org/advisories/CA-2001-26.html`, September 2001.

[3] "CERT Advisory CA-2001-19: 'Code Red' Worm Exploiting Buffer Overflow in IIS Indexing Service DLL," `http://www.cert.org/advisories/CA-2001-19.html`, July 2001.

[4] "Cert Advisory CA-2003-04: MS-SQL Server Worm," `http://www.cert.org/advisories/CA-2003-04.html`, January 2003.

[5] "The Spread of the Sapphire/Slammer Worm," `http://www.silicondefense.com/research/worms/slammer.php`, February 2003.

[6] Colleen Shannon and David Moore, "The Spread of the Witty Worm," *IEEE Security & Privacy*, vol. 2, no. 4, pp. 46–50, July/August 2004.

[7] S. E. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specifi cation," RFC 2460, Internet Engineering Task Force, Dec. 1998.

[8] Eugene H. Spafford, "The Internet Worm Program: An Analysis," Tech. Rep. CSD-TR-823, Purdue University, 1988.

[9] "Google, other engines hit by worm variant," `http://www.technewsworld.com/story/35351.html`, July 26, 2004.

[10] S. Staniford, V. Paxson, and N. Weaver, "How to Own the Internet in Your Spare Time," in *Proceedings of the* 11^{th} *USENIX Security Symposium*, August 2002, pp. 149–167.

[11] C. C. Zou, W. Gong, and D. Towsley, "Code Red Worm Propagation Modeling and Analysis," in *Proceedings of the* 9^{th} *ACM Conference on Computer and Communications Security (CCS)*, November 2002, pp. 138–147.

[12] Cliff Changchun Zou, Weibo Gong, and Don Towsley, "Code Red Worm Propagation Modeling and Analysis," in *Proceedings of the* 9^{th} *ACM conference on Computer and Communications security*, 2002.

[13] F. Cohen, "Computer Viruses: Theory and Practice," *Computers & Security*, vol. 6, pp. 22–35, February 1987.

[14] Jeffrey O. Kephart, "A Biologically Inspired Immune System for Computers," in *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*. 1994, pp. 130–139, MIT Press.

[15] C. Cowan, C. Pu, D. Maier, H. Hinton, J. Walpole, P. Bakke, S. Beattie, A. Grier, P. Wagle, and Q. Zhang, "StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks," in *Proceedings of the* 7^{th} *USENIX Security Symposium*, January 1998.

[16] Hao Chen, Drew Dean, and David Wagner, "Model Cehcking One Million Lines of C Code," in *Proceedings of the Network and Distributed System Security (NDSS) Symposium*, February 2004, pp. 171–185.

[17] K. Lhee and S. J. Chapin, "Type-Assisted Dynamic Buffer Overflow Detection," in *Proceedings of the* 11^{th} *USENIX Security Symposium*, August 2002, pp. 81–90.

[18] E. Larson and T. Austin, "High Coverage Detection of Input-Related Security Faults," in *Proceedings of the* 12^{th} *USENIX Security Symposium*, August 2003, pp. 121–136.

[19] D. Moore, C. Shanning, and K. Claffy, "Code-Red: a case study on the spread and victims of an Internet worm," in *Proceedings of the* 2^{nd} *Internet Measurement Workshop (IMW)*, November 2002, pp. 273–284.

[20] D. Song, R. Malan, and R. Stone, "A Snapshot of Global Internet Worm Activity," Tech. Rep., Arbor Networks, November 2001.

[21] C. C. Zou, L. Gao, W. Gong, and D. Towsley, "Monitoring and Early Warning for Internet Worms," in *Proceedings of the* 10^{th} *ACM International Conference on Computer and Communications Security (CCS)*, October 2003, pp. 190–199.

[22] T. Liston, "Welcome To My Tarpit: The Tactical and Strategic Use of LaBrea," `http://www.threenorth.com/LaBrea/LaBrea.txt`, 2001.

[23] M. Williamson, "Throttling Viruses: Restricting Propagation to Defeat Malicious Mobile Code," Tech. Rep. HPL-2002-172, HP Laboratories Bristol, 2002.

[24] J. Twycross and M. M. Williamson, "Implementing and testing a virus throttle," in *Proceedings of the* 12^{th} *USENIX Security Symposium*, August 2003, pp. 285–294.

[25] C. Wong, S. Bielski, J. M. McCune, and C. Wang, "A Study of Mass-Mailing Worms," in *Proceedings of the ACM Workshop on Rapid Malcode (WORM)*, October 2004, pp. 1–10.

[26] D. Moore, C. Shannon, G. Voelker, and S. Savage, "Internet Quarantine: Requirements for Containing Self-Propagating Code," in *Proceedings of the IEEE Infocom Conference*, April 2003.

[27] C. Wang, J. C. Knight, and M. C. Elder, "On Computer Viral Infection and the Effect of Immunization," in *Proceedings of the* 16^{th} *Annual Computer Security Applications Conference (ACSAC)*, 2000, pp. 246–256.

[28] C. Nachenberg, "Computer Virus - Coevolution," *Communications of the ACM*, vol. 50, no. 1, pp. 46–51, 1997.

[29] M-J Lin, A. Ricciardi, and K. Marzullo, "A New Model for Availability in the Face of Self-Propagating Attacks," in *Proceedings of the New Security Paradigms Workshop*, November 1998.