# Simulating Realistic Network Worm Traffic for Worm Warning System Design and Testing

Michael Liljenstam David M. Nicol Vincent H. Berk Robert S. Gray

{mili,nicol,vberk,rgray}@ists.dartmouth.edu
Institute for Security Technology Studies
Dartmouth College
45 Lyme Rd., Suite 300
Hanover, NH 03755

## ABSTRACT

Reproducing the effects of large-scale worm attacks in a laboratory setup in a realistic and reproducible manner is an important issue for the development of worm detection and defense systems. In this paper, we describe a worm simulation model we are developing to accurately model the large-scale spread dynamics of a worm and many aspects of its detailed effects on the network. We can model slow or fast worms with realistic scan rates on realistic IP address spaces and selectively model local detailed network behavior. We show how it can be used to generate realistic input traffic for a working prototype worm detection and tracking system, the Dartmouth ICMP BCC: System/Tracking and Fusion Engine (DIB:S/TRAFEN), allowing performance evaluation of the system under realistic conditions. Thus, we can answer important design questions relating to necessary detector coverage and noise filtering without deploying and operating a full system. Our experiments indicate that the tracking algorithms currently implemented in the DIB:S/TRAFEN system could detect attacks such as Code Red v2 and Sapphire/Slammer very early, even when monitoring a quite limited portion of the address space, but more sophisticated algorithms are being constructed to reduce the risk of false positives in the presence of significant "background noise" scanning.

## Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Modeling techniques—simulation; C.2.0 [**Computer and Communication Networks**]: Security and Protection—worms; C.2.3 [**Network Operations**]: Network monitoring—worm detection; C.2.5 [**Local and Wide-Area Networks**]: Internet

## General Terms

Security, Experimentation, Measurement, Performance


*WORM'03,* October 27, 2003, Washington, DC, USA.


## Keywords

Network Security, Network Modeling and Simulation, Worms, Worm Detection Systems, Code Red, Slammer

## 1. INTRODUCTION

Network worm infestations over the last few years, such as Code Red [16, 22], Code Red II [16, 22], and Nimda [22] in 2001, and recently the Sapphire/Slammer [15] worm have focused a lot of attention on threats posed by self-replicating malicious code. As a result, efforts are under way to create early warning systems and various defense mechanisms, e.g. [5, 26, 11, 1], and the general feasibility of such efforts is the focus of studies such as [17, 6].

However, since large-scale worm events are fortunately still rare, it is not possible to test prototype designs in a live setting. Moreover, faithfully reproducing the effects of large-scale worm attacks in a laboratory setup is difficult. In this paper we describe a worm simulation model we are developing, and how it is being used to generate test data for the Dartmouth DIB:S/TRAFEN System [5]. This particular worm detection system collects copies of ICMP messages generated by random scanning and tries to recognize signatures of early worm propagation by correlating the collected messages. Questions regarding things like necessary collection point coverage, "signal-to-noise" ratio for reliable detection, and optimal parameter settings in the correlation algorithms are all difficult to answer without some way to generate realistic test traffic.

Accurately simulating the effects that large-scale worm infestations have on infrastructure is challenging since, in addition to issues related to heterogeneity and change in the Internet [12]: *i)* a worm that infects tens or hundreds of thousands of machines on the Internet gives rise to an *inherently large-scale phenomenon*, and requires the model to be of appropriate scale to correctly model the propagation dynamics; *ii)* with few exceptions, most worms have propagated over time scales of hours to days (or longer), thus it may result in a *large span of timescales* where network events at timescales down to microseconds are simulated over days. This could be further complicated in the case of "stealthy worms" propagating slowly to avoid detection.

For a credible model we want the worm spreading through a network with a *realistic number of hosts* using *realistic scan*

*rates* over a *realistic address space*. This will ensure that the propagation dynamics are realistic and will avoid artifacts as we extract more detailed information from the model. The model we describe combines modeling at multiple levels of abstraction in order to be both detailed enough to generate realistic packet traffic, and efficient enough to model a worm spreading through the Internet. We are developing this model with the aim to provide a general testbed for worm detection and countermeasure systems. It is currently under active development and is being publicly released as an add on package (called `SSF.App.Worm`) to the SSFNet simulator [2][9].

The remainder of this paper is organized as follows: We introduce the DIB:S/TRAFEN system in Section 2, the simulation model in Section 3–4, and validate the model in Section 5. Section 6 describes our case study, where we use the simulator to test the detection algorithms currently implemented DIB:S/TRAFEN. We discuss related work in Section 7, and finally conclude in Section 8.

## 2. DIB:S/TRAFEN OVERVIEW

The focus of this paper is the worm simulation model, but we start by introducing the DIB:S/TRAFEN system [5], which sets the scene for the simulation and the case study presented later. DIB:S/TRAFEN can detect and classify active Internet worms in their earliest stages of propagation, increasing the chance of intervention and subsequent mitigation of Internet-scale epidemics.

Most current active worms spread by randomly probing IP addresses and, since the IPv4 address space is densely populated, even unbiased random scanning will find vulnerable systems relatively quickly. This random scanning however, will probe many unassigned IP addresses, i.e., addresses that are not associated with a reachable computer. In many cases, routers that receive a packet destined for an unreachable IP address will drop the packet and return an *ICMP Destination Unreachable* (ICMP Type 3) message to the packet originator. This ICMP-T3 message will include the original IP header and at least 8 bytes of the protocol header, which together will include the source and destination IP addresses and port numbers for both UDP and TCP packets. This embedded data makes ICMP-T3 messages very useful for detection of scanning events, and, in fact, our DIB:S system collects these messages by having a select group of participating routers forward all the ICMP-T3 messages that they generate to an analysis station. The generation of ICMP-T3 messages is rate limited, usually at 3 per second, and will add negligible overhead to the router.[1] In addition, if site policy so dictates, the ICMP-T3 message can be sent to the analysis station, but not sent to the scanning machine, preventing easy reconnaissance of the network. Finally, although we will see later that the total number of participating routers can be small, these routers must be distributed across a significant fraction of the Internet address space to ensure timely and accurate worm detection.

[1]Note that routers usually apply this rate limit to all outgoing ICMP messages, rather than to individual ICMP message subtypes or network interfaces. Traffic, on any interface, that generates ICMP messages will reduce the number of worm-related ICMP-T3 messages sent to the analysis station. Initial experience indicates that this bias has only a modest effect across a group of instrumented routers.

As the ICMP-T3 messages arrive at the DIB:S analysis station, they are sorted and analyzed according to the embedded source and destination addresses and ports. When the total number of packets for any source or destination machine or port exceeds a threshold $N_{DIBS}$ within a configurable time interval $\Delta t_{DIBS}$, DIB:S generates a scan alert, and DIB:S will not generate another scan alert for the same machine and port combination until $\Delta t_{DIBS}$ seconds elapse. DIB:S generates several types of scan alert, corresponding to a single source machine, multiple source machines, and so on, but the most important scan alert for worm detection is when a single source machine uses the same protocol $P$ to contact the same port $p$ on $N_{DIBS}$ target machines within $\Delta t_{DIBS}$ seconds. This alert indicates the "bloom" typical of random scanning behavior originating from a single host, and an exponential increase in the number of alerts for the same port and protocol most likely indicates a propagating worm.

It is the job of TRAFEN to detect this exponential increase. TRAFEN, which stands for TRacking And Fusion ENgine and is an implementation of a *Process Query System* [4], is a domain-independent middleware that allows the rapid development of tracking and data-fusion applications. The developer defines a process model that describes how to identify which incoming observations correspond to the *same* real-world process, such as a propagating worm or a vehicle moving through a battlespace, and how to predict the future state of a previously identified process. The process model can be defined in many ways, such as Hidden Markov Models [19], Kalman filters, or domain-specific rulebases. TRAFEN uses traditional tracking algorithms, most notably an implementation of Reid's multiple hypothesis tracking (MHT) algorithm [20], to handle the mechanics of constructing the most likely observation groupings. Specifically, TRAFEN keeps multiple *hypotheses*, where each hypothesis is a set of *tracks* of related observations. Each observation is represented only once in each hypothesis, and each hypothesis aims to represent an accurate view of the world. Using the process model, each incoming observation is compared with each track in each hypothesis, and one or more new hypotheses, each containing the new observation, are generated for each existing hypothesis. The hypotheses are ranked according to their likelihood, and then are pruned to prevent an exponential increase as further observations arrive. In our case, the observations are DIB:S alerts, and our current working prototype uses a simple rule-based process model to identify worm activity.

Collecting ICMP-T3 messages from instrumented routers, rather than using ingress and egress filters at network boundaries, allows efficient detection of scans that span multiple networks, even if the scan probes only one address from each network. Moreover, using instrumented routers associated with populated address space, rather than only with entirely unallocated address blocks, makes it harder for worm authors to avoid the detection system. DIB:S and TRAFEN could use multiple types of scan data to improve their detection performance, however.

## 3. THE WORM SIMULATION

For the worm simulation, our starting point is the SSFNet simulator [2][9], a packet-level network simulator written in Java that supports parallel and distributed execution for increased scalability. It includes standard TCP/IP protocols

(IP, ICMP, TCP, UDP, HTTP, ...) and detailed implementations of routing protocols such as BGP and OSPFv2.

The challenge of simulating worm events involving as many as hundreds of thousands of hosts generating high rates of scan packets can quickly lead to resource and performance demands that are beyond even state of the art packet-level simulators running on super-computers. Consider, for instance, the Slammer worm [15], which infected at least 75,000 hosts with an estimated average scan rate of about 4000 scans/s per worm.[2] Then, if we assume this scan rate at the peak of infection, we would have to simulate $75,000 \cdot 4000 \cdot h = 3 \cdot 10^8 \cdot h$ *Packet Transmission Events* per second of simulated time, where $h$ is the mean number of hops a scan packet travels. This, and the memory and effort required to model significant portions of the Internet at this level of detail, has led us to explore a simulation that mixes modeling at dual levels of abstraction.

The propagation dynamics of worms spreading by uniform random scanning, such as Code Red, lend themselves well to a coarse form of modeling based on epidemic models. This drastically reduces resource requirements compared to a packet-level model of the whole system. On the other hand, epidemic models alone include no information about the underlying network and its possible effects on the propagation, e.g., bandwidth constraints on scans as observed during Slammer or router failures observed due to worm traffic [24, 8]. Moreover, additional details are required to create meaningful, realistic test data for detection systems. Consequently, we have chosen to combine a coarse-level model of the worm propagation and scan traffic with detailed models (packet-level simulation) of selected parts of the network. More details on the benefits achieved and trade-offs involved in this approach can be found in [14] where we proposed the mixed abstraction-level model.

Figure 1 illustrates how a coarse "*macroscopic model*", such as an epidemic model of the worm spreading, drives the network model in terms of host infections and scan traffic induced in the network. In the simplest case, we assume that the worm spread is not affected by what goes on at the network level, although other "flavors" of macroscopic models also allow us to take network effects (such as failures, routing dynamics, or congestion) into account.

## 3.1 Homogeneous Deterministic Epidemic

The simplest form of macroscopic model implemented in the simulator is the continuous-time deterministic version of the "general epidemic model" [10]. The model assumes a fixed population of size $N$ and describes the evolution of the system through a set of equations:

$$\frac{ds(t)}{dt} = -\beta s(t) i(t) \quad (1)$$

$$\frac{di(t)}{dt} = \beta s(t) i(t) - \gamma i(t) \quad (2)$$

$$\frac{dr(t)}{dt} = \gamma i(t) \quad (3)$$

where the constant $\beta$ is the *infection parameter*, i.e., the pairwise rate of infection, and the constant $\gamma$ is the *removal parameter*. These equations describe the rate of transitions from the population of susceptible hosts $s$ to the infected population $i$ to the removed population $r$. Thus, $s(t)+i(t)+r(t) = N, \forall t \geq 0$. Here we assume that the total population under consideration is large enough so that mean evolution of the stochastic system can be approximated using this "law of mass action"-formulation, and we assume that all (susceptible) hosts in the Internet can be viewed as one homogeneous population. Furthermore, it assumes *homogeneous mixing*—meaning roughly that interaction is equally likely between all members of the population in a given (small) interval of time.

A similar model appears to first have been proposed for the Code Red worm in [21], and epidemic models have since been considered in several studies [22, 14, 16, 26, 17].

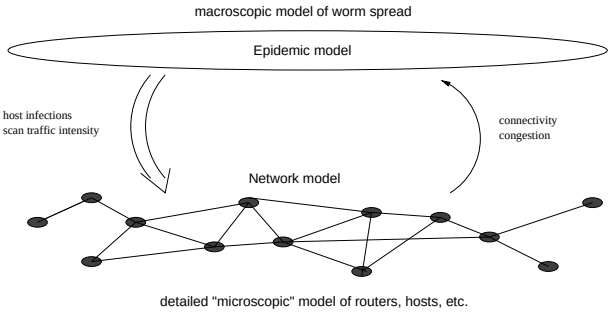


**Figure 1: The simulation uses models at two different levels of abstraction: a significant fraction of the Internet is modeled coarsely at the "macroscopic" level, and selected parts are modeled using full packet level simulation. The coarse model of the worm infection drives the network model, and some (limited) feedback may occur, such as accounting for changes in connectivity.**

## 3.2 Stochastic Epidemic (Homogeneous)

For smaller populations or early stages of propagation, it is important to model the stochastic evolution of the system for better fidelity. Consider a worm spreading by sending scans (infection packets) at random with uniform distribution over the 32 bit IPv4 address space. Thus, a scan from an infected host will hit a specific address with probability $P[hit] = 2^{-32}$.[3] If $\sigma$ is the scan rate (to unique addresses) of a single worm, then $i$ infected hosts generate $\sigma \cdot i$ scans. Thus, if $s$ is the number of susceptible hosts, then the number of new infections $X$ in a time interval $\Delta t$ has a Binomial distribution with parameters $\text{Bin}(s, p)$ where $p = P[hit] \cdot i \cdot \sigma \cdot \Delta t$. If $X_t$ is the number of infections at time $t$ and $Y_t$ is the number of removals (similarly defined), then a discrete time model of the system can be written as:

$$s(t+\Delta t) = s(t) - X_t \quad (4)$$

$$i(t+\Delta t) = i(t) + X_t - Y_t \quad (5)$$

$$r(t+\Delta t) = r(t) + Y_t \quad (6)$$

Since $X$ has a Binomial distribution, $E[X] = s \cdot p = s \cdot P[Hit] \cdot i \cdot \sigma \cdot \Delta t = \{\text{regrouping}\} = (P[Hit] \cdot \sigma \cdot \Delta t) \cdot s \cdot i$. Comparing back to Equation 1, it is easy to see how

[2]This scan rate was observed early, before bandwidth limitations set in. Thus, at the peak of infection, it should have been somewhat lower.

[3]This is slightly simplified as it ignores reserved address space for multicast and loopback, but taking these address blocks into account makes no significant difference.

the deterministic model describes the mean evolution of the system and that $\beta = \sigma \cdot \Delta t \cdot P[Hit]$. Note also that it is the uniform distribution of the scans that ensures that the homogeneous mixing assumption is true. For instance, it does not hold for viruses propagated by emails or magnetic media since these interactions are typically constrained by human social relationships [13].

## 3.3 Spatial Epidemic

To study scan traffic flows, we break the model down spatially by using the 'stratified population' formulation of the epidemic model [10], where (in this case) the host population is stratified by network:

$$\begin{aligned}
\frac{ds_j(t)}{dt} &= -s_j(t) \cdot (\beta_{1j} i_1(t) + \ldots + \beta_{mj} i_m(t)) \\
\frac{di_j(t)}{dt} &= s_j(t) \cdot (\beta_{1j} i_1(t) + \ldots + \beta_{mj} i_m(t)) - \gamma_j i_j(t) \\
\frac{dr_j(t)}{dt} &= \gamma_j i_j(t)
\end{aligned}$$

where, for each population group $j$: $s_j(t)$, $i_j(t)$, and $r_j(t)$ are the state variables, $\gamma_j$ is the removal parameter, and $\beta_{ij}$ is the infection parameter between groups $i$ and $j$.

For a system that behaves as the homogeneous model, we set $\beta_{ij} = \beta$, since this gives us (ignoring removals for clarity)

$$\sum_j \frac{di_j(t)}{dt} = \beta \cdot i(t) \cdot \sum_j s_j(t) = \beta \cdot s(t) \cdot i(t)$$

i.e., the same infection growth rate as the homogeneous case. A corresponding stochastic model can also easily be created, as in the previous section.

## 3.4 Scan Traffic

Given a spatial model of the spread, we can easily calculate the mean traffic intensity for egress and ingress scans to each network. We have scans generated from network $j$ at rate $\sigma_j^{\text{gen}}(t) = i_j(t) \cdot \sigma$ and scans destined for network $j$ at rate $\sigma_j^{\text{dest}}(t) = i(t) \cdot \sigma \cdot \frac{A_j}{2^{32}}$, where $A_j$ is the size of network $j$'s address space. Hence, for network $j$ we have egress scans $\sigma_j^{\text{egr}}(t) = \sigma_j^{\text{gen}}(t) \cdot (1 - \frac{A_j}{2^{32}})$ and ingress scans $\sigma_j^{\text{ingr}}(t) = \sigma_j^{\text{dest}}(t) - \sigma_j^{\text{gen}}(t) \cdot \frac{A_j}{2^{32}}$. This simple model is sufficient for studying scan traffic passing through gateway routers of "edge" networks if we assume networks with a single gateway. Moreover, if we assume that we study ASes and let each network in our model be an AS, the model is simple enough to allow us to simulate the entire AS-topology of the Internet.

Routers covering "edge networks" are expected to be useful detectors as they will generate ICMPs for unreachable hosts. Core backbone routers could be useful for covering more network traffic, but on the other hand can only "detect" traffic going to bogus (unadvertised) prefixes. Thus, in the following, we will model the effect of equipping edge networks gateway routers with ICMP BCC: capabilities for the DIB:S/TRAFEN system. Note that this means we do not need to model the traffic flows between ASes or routers "internally" in the AS topology. Hence, the simulator loads an AS-topology model, based on adjacencies from a BGP table dump. We stress, however, that it is only the size of the networks that is relevant here.[4]

[4]We thus avoid issues of incompleteness of the AS-graph and routing policies that constrain possible traffic flows.

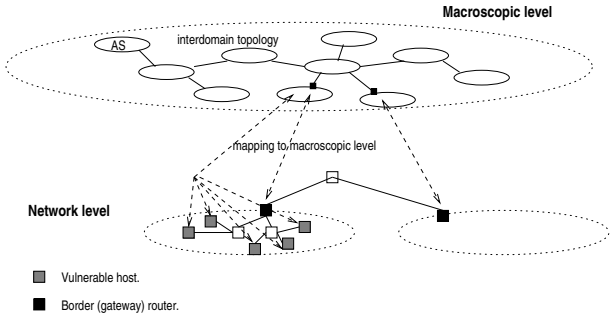


**Figure 2: Certain key entities in the detailed network model, such as vulnerable hosts and gateway routers, can be mapped to corresponding entities at the macroscopic level.**

The worm-modeling framework allows the user to map certain entities in the network model, i.e., elements represented at full packet-level detail using standard SSFNet constructs, to corresponding entities in the macroscopic model. Thus, it is possible to model the whole Internet (or a large part of it) coarsely, and selected parts in more detail, as illustrated in Figure 2. We return later to how this modeling pattern is used to generate packet-level information from the macroscopic model.

In this study, we simulate two observed worms, Code Red v2 (CRv2) on July 19, 2001 [16, 22] and Sapphire/Slammer (Slammer) on Jan 25, 2003 [15], to demonstrate the validity of the model. It is also possible to generalize over the parameter space of plausible hypothetical worms, however. The parameter choices and other specifics for the CRv2 and Slammer worm simulations will be described later in Section 5 where we also validate these models by comparing them to real captured traffic during the attacks.

# 4. GENERATING TEST TRAFFIC

As illustrated in Figure 3 the DIB:S/TRAFEN system operates by collecting copies of ICMP type 3 (unreachable) messages. Instrumented routers in the Internet send copies of ICMP type 3 messages to the DIB:S system which correlates and analyzes the data.

We simulate AS networks at the macroscopic level, that is, abstracting away internal details. A subset of the ASes are assumed to have a single gateway router which is instrumented to send ICMP copies. These instrumented routers are modeled at the detailed (packet) level and mapped to the macroscopic level as was shown in Figure 2. The DIB:S collection point host is also modeled at the detailed (packet) level, residing in a separate AS.

In this study we focus on ingress scans observed at the instrumented gateway routers and do not generate ICMPs for any egress scans that might be coming from within the router's network. The instrumented router model has to translate the observed ingress scan rate into a packet process and generate ICMP packets corresponding to observed scans that target an address where there is no host. As shown in Figure 3, all packets arriving to the DIB:S system in the model are dumped to a file in binary `tcpdump` format. Using the `tcpreplay` tool [23], we can then replay the packet stream into the real DIB:S system to simulate

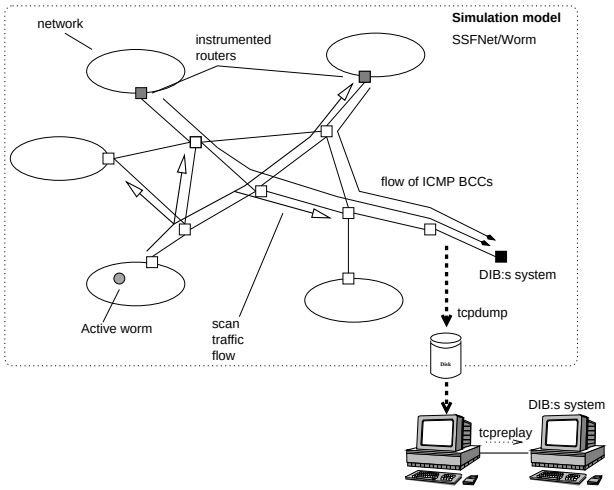


**Figure 3: Simulation of worm and DIB:S system. Instrumented routers send "Blind Carbon Copies" of generated type 3 ICMP messages to the DIB:S system. The system analyzes the ICMPs to identify scanning activity, and correlates that scanning activity to track worm infection. The worm and collection system are simulated, and the ICMP copies arriving at the DIB:S system are dumped to file and then replayed against the real DIB:S system host.**

the ICMP packets observed during the attack. However, DIB:S is also able to directly read `tcpdump` files for running experiments faster than real time.

*From Flows to Packets.* The ingress scan flows are modeled (at the macroscopic level) as a piecewise constant flow rate. At the instrumented routers, these flows must be converted into a packet arrival process, and because we are adding information, any such conversion will, by necessity, be an approximation. Since the incoming scans at a router are generated independently by many sources, we have chosen (based solely on this argument) to model the arrivals of scans (for distinct destination IP addresses) as a Poisson process. At the beginning of each time-step interval the arrival rate is set to the ingress scan flow rate $\sigma_j^{\text{ingr}}(t)$. Inter-arrival times are sampled and the next packet arrival determined. However, if the next packet arrival falls beyond of the current time-step it is discarded and a new sample taken at the beginning of the next time-step. This prevents prolonged quiet periods as the scan rates gradually grow from very low levels and will accurately model a Poisson process once $\sigma_j^{\text{ingr}}(t) \cdot \Delta t \gg 1$.

If the worm uses a transport protocol that generates retry packets, e.g. CRv2 where TCP generates SYN retries, those packets are added. Captured CRv2 packet traces indicate that the Microsoft TCP implementation generates a retry 3 seconds after the initial SYN, and a second retry 6 seconds after the first retry. These retries are scheduled accordingly into the packet arrival process when TCP is simulated.

For each network $j$ a host address utilization fraction $u_j$ is configured, denoting the fraction of addresses occupied by hosts. A Bernoulli trial process with probability $1 - u_j$ determines if an ICMP should be generated for an arriving scan packet.

Finally, RFC 1812 [3] states that routers should be able to rate limit the generation of ICMP messages to reduce load. We have observed that this is commonly used, with typical rate limits in the range of 3–4 ICMPs per second. A rate limit of 3 ICMPs/s was implemented in the simulator and was used in the experiments.

*Generating ICMP Packets.* Certain ICMP packet content is used by the DIB:S system correlation, and thus has to be generated with some care:

**IP header** SSFNet generates the IP header. The simulated source and destination addresses are replaced by real source and destination addresses as the packets are dumped to the packet trace file. This way the convenience of SSFNet's automatic IP address assignment is not sacrificed.

**ICMP header** We let ingress scans result in type 3 subtype 1 "host unreachable" packets.

**Embedded headers (IP/TCP/UDP)** Each host infected by the worm is assigned a unique IP address sampled uniformly over the address space assigned to the network in which the host resides. The source IP address is drawn with equal probability among all active worm copies. The destination IP address is drawn uniformly from the destination network address space. The destination port number is known for the worm in question, and the source port is drawn with uniform distribution from ports above the "well-known" range, i.e., from 1024–65535.

*Background Noise.* It is useful to also be able to generate "background scanning noise" for at least two reasons: *i)* to model cases, such as for the CRv2 worm, where there was a significant level of scans going on before the worm was launched, and *ii)* to test what "signal-to-noise ratio" is necessary for reliable detection. The scans before CRv2 were partly due to version 1 of the worm (that was ineffective in spreading) and partly because TCP port 80 (HTTP) is simply a popular port to probe for vulnerabilities.

We have included the option of adding background noise with a fixed rate of scans in the simulator. Again, since we expect that the scans are generated by a large number of independent sources, we model the arrivals using a Poisson process, which is simply superimposed on other scan arrivals. For background scans, the source IP addresses are expected to lack time locality, so they are sampled uniformly over the whole IPv4 address space.

# 5. MODEL PARAMETERS AND VALIDATION

We validate our simulation by comparing output from the model with real captured scan traces from the Code Red v2 and Slammer attacks. The simulation model used for this comparison is the spatially distributed deterministic model where $\beta$ is calculated from the scan rate. It is more convenient to use the deterministic version of the model for comparisons with real data since the infection 'takes off' at a predictable time point. However, *for the actual testing, we use the stochastic model* as it captures the important variability inherent in propagation from small infected populations.

*Code Red v2.* Using scan traffic data collected by the Chemical Abstract Service (CAS) during the Code Red v2 attack, we estimate the number of infected hosts over time using the bias adjustment method proposed by Zou et al. [27]. Figure 4 shows counts of unique source IP addresses in scans

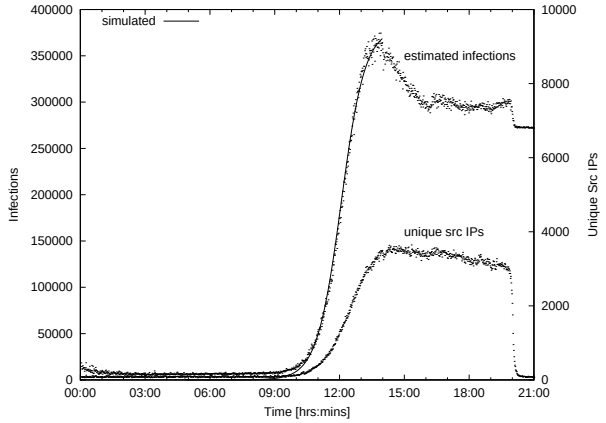


**Figure 4: Raw counts of unique scan source IPs captured at CAS for Code Red v2 on July 19th, 2001, in 1 minute bins are shown (with respect to the right-hand y-axis). Also shown is a comparison of estimated infections based on the source IP counts (dots) and infections in the simulation (line).**

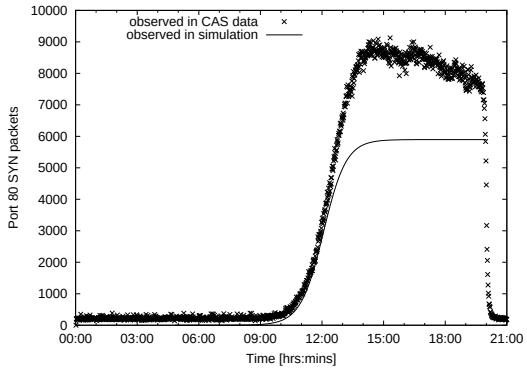


**Figure 5: Comparison of total number of SYN packets captured in one minute bins on the CAS network during CRv2 with observations on a /16 network in the simulation.**

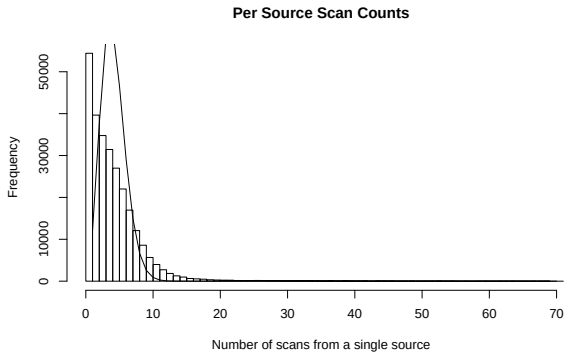


**Figure 6: Histogram of the number of scans (excluding SYN retries) from each source IP within the 10 hour attack period compared to the expected Binomial distribution.**

destined for the CAS /16 network on TCP port 80 (on the right hand side y-axis), and it shows the estimated global number of infected hosts after bias adjustment (on the left hand axis).[5] Since our primary interest is the initial growth of infection, we do not model host patching or filtering and set $\gamma = 0$, and, as the attack occurred over the span of several hours we use a fairly coarse time step $\Delta t = 60$ seconds.

These estimates indicate at least 360,000 susceptible hosts (which agrees well with [16]), and running Code Red v2 in a lab setup we have observed a mean scan rate slightly higher than 5 scans per second. Based on the estimated infections, we set the population of susceptibles $N = 380,000$ and the scan rate $\sigma = 5.65$ scans/s, since these produce a reasonably good fit to the growth phase of the infection as shown with the solid line in Figure 4.

It can be noted that whereas the simulation starts off with a single infection, the estimates start at a higher level. As was mentioned in Section 4, the CRv2 attack was preceded by a significant level of background noise, in part due to an earlier version of Code Red (v1) that was released earlier, but was less effective in spreading.

As a validation step, we pick a /16 network in the model (same size as the CAS network) and count all incoming scan SYN packets, just like the raw data for the real CAS network. The result is shown in Figure 5 and reveals a puzzling aspect of the real captured data set: at the peak of infection, the real data contains an almost 50% higher scan packet rate than would be expected.[6]

Essentially, there is a discrepancy between the observed number of unique source addresses in the scans and the total number of scans in the captured data. To understand this better, we examined the packet trace and counted the number of scans received from each source IP address. Figure 6 shows a histogram of the scans per source count. During the 10 hour attack period, the trace contains 269,695 unique sources sending 1,325,753 SYNs in total. The graph also shows a Binomial distribution where it is assumed that all sources were active for the entire period. Since many sources started scanning at some later point, there will tend to be more low counts than predicted by the distribution, as can be seen in the graph. However, the histogram also shows more mass in the tail than expected, i.e., many sources sending an unexpectedly high number of scans. In fact, whereas the expected number of scans from a single source in this time period is approximately 3.1, one source sent as many as 251 scans to this network.

Based on this, we conclude that the trace contains a higher scan rate per source than expected from scan rate exper-

[5]The bias adjustment essentially attempts to infer the true number of active worms based on the cumulative count and difference in numbers of unique scan source IPs observed on a limited address space.

[6]A simple calculation shows that the simulator produces the expected result given the assumptions: At the peak of infection, $i$ is close to 380,000, and $\sigma = 5.65$ scans/s. Let $E[X]$ be the expected number of scan packets observed during a one minute interval. Then $X \in \text{Bin}(n, p)$ with $n = i \cdot 3\sigma$ and $p = 2^{-(32-16)}$. Thus, at the peak of infection, $E[X] = n \cdot p \approx 5900$ scans/min.

iments and a uniform scan distribution. There could be many possible explanations for this, from imperfections in the worm's random number generation to possibilities of Network Address Translation gateways or reinfections. Ultimately, this discrepancy will not affect our tests significantly, as we are testing detection during the early stages of growth where the model corresponds well with reality. Thus, we prefer to use this model, which is well understood, rather than adding any ad-hoc terms to account for the difference.

*Sapphire/Slammer.* We base our model of the Slammer worm largely on the analysis in [15] and a data set we have obtained from TRIUMF Canada. This worm spread much faster than CRv2 and is estimated to have infected most of the vulnerable hosts within 10 minutes. Modeling this worm also involves a few more complications: Firstly, the scan rate was essentially *bandwidth constrained* which affected the propagation speed and possibly also the observations on a single network (i.e., the incoming scan traffic that could be observed is also likely to be limited at some point). Secondly, code analysis in [15] indicated that it had faulty random number generator code. Thus, some address spaces would never be scanned by a single worm, although a large number of worms are likely to jointly cover the entire address space. The net effect is that collected data sets from any single site must be viewed with some caution, especially for a small number of worms. We have included facilities to model access-link bandwidth limitations in the simulator, but found that the worm detection point (the feature we want to test in this study) occurs so early on that the bandwidth constraints can be ignored in the model. We model the worm scanning using uniform distribution over the entire IPv4 space (not taking the defects into account) and thus effectively assume that any measurements taken in the model are from networks that would not have been bypassed by the worm scans. Because of the speed of the worm, we use a smaller time step $\Delta t = 1$ second, and again since our primary interest is in the initial stages of worm spread, we set $\gamma = 0$ (no patching or filtering).

Using TRIUMF Canada data, we estimated the number of infected hosts as the Slammer worm spread. We expect to see exponential growth in the early stages before bandwidth limitations set in. Figure 7, top graph, shows a semilog plot of the first two minutes of the worm data where there appears to be a short straight line segment, indicating exponential growth. [15] states that by observing data from multiple networks, they found at least 75,000 infected hosts, an average scan rate $\sigma \approx 4000$ scans/s, and $7 \pm 1$ new infections/min by a single worm during early spread. At startup from a single infection, this means $i(0) = 1$ and $\frac{di(t)}{dt}|_{t=0} = \beta s(0) i(0) = \beta s(0) = 7 \pm 1$ infections per minute, with $\beta = \sigma \cdot \Delta t \cdot P[Hit] \approx 5.6 \cdot 10^{-5}$. This would mean $s(0)$ in the range of 107,000–143,000. We set $N = 120,000$ resulting in a fairly good fit for the first two minutes of propagation, as also shown in the top graph of Figure 7. The bottom graph in Figure 7 shows the same plot with linear axes where it is more evident how the infection growth rate slows down compared to the model. However, the early exponential growth stage is the time period that is relevant for the tests performed in this study and during that stage the model matches reality well.

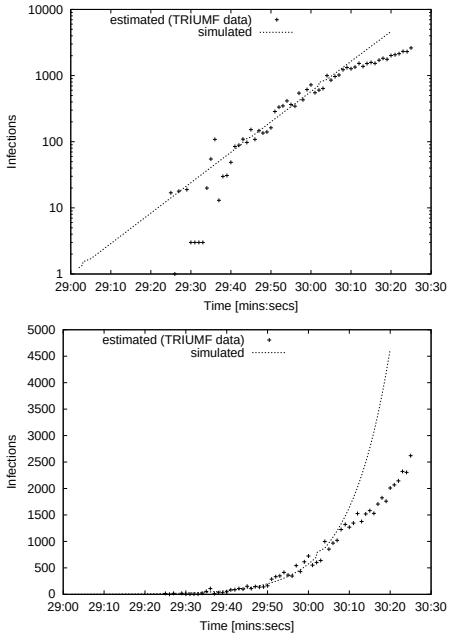


**Figure 7: The first couple of minutes of the Sapphire/Slammer attack as captured at TRIUMF Canada. The graphs show estimated number of infections (calculated from unique source IPs) compared to a simulation of unconstrained spread (no bandwidth constraints). The top graph is a semilog plot and the bottom graph uses linear scale.**

## 6. EXPERIMENTS

We evaluated the performance of DIBS/TRAFEN by running simulations with varying degrees of router participation, feeding the simulated ICMP-T3 Unreachable traffic into the working DIBS/TRAFEN prototype, and measuring how many simulated machines were infected at the time that DIBS/TRAFEN detected the propagating worm. Figure 8a shows the results for a simulated Code Red v2 worm, while Figure 8b shows the results for a simulated Sapphire/Slammer worm.[7] For the Code Red worm, we did one set of runs with background scanning noise and one set of runs without noise. Due to time constraints with the Slammer worm, we did only a set of runs without noise, but the Code Red noise results generalize to the Slammer case. For all runs, the simulation parameters were the same as in the validation experiments. Specifically, the address space was the Internet or IPv4 address space of $2^{32}$ addresses; the simulated Code Red v2 and Sapphire/Slammer worms scanned at rates of 5.65 and 4000 scans per second respectively; there were 384,000 and 120,000 vulnerable machines respectively; the address space utilization, or percentage of reachable addresses, in each of the detector networks was 50%; and the background

[7]We did not perform experiments with the real-world Slammer and Code Red v2 datasets obtained by different organizations at their individual network boundaries, since most of this data does not allow an easy mapping to ICMP-T3 messages. In addition, ICMP-T3 messages from only a single site generally will not provide good detection performance, although we do hope to combine some of the real-world datasets for future experiments.

scanning rate, for the one Code Red experiment, was 1.41 coincidental probes to the Web service port *per Class B network per second* (as observed in the CAS data). We selected a 50% reachability, which is significantly higher than most real Class B networks, to be conservative and ensure that our evaluation did not produce more ICMP-T3 messages than would be observed on the real Internet.

For DIBS, the relevant parameters are $N_{DIBS}$ and $\Delta t_{DIBS}$, the number of ICMP-T3 messages per scan alert and the size of the history window respectively. A lower value of $N_{DIBS}$ increases the chances of false positives, since ICMP-T3 background noise can propagate into the scan alerts, and any value below $N_{DIBS} = 4$ makes the system unusable. Conversely, although higher values will lead to more accurate detection, the moment of detection will be later, possibly *too* late. Initial experimentation has shown that $5 \leq N_{DIBS} \leq 15$ gives the best results, and that the value of $N_{DIBS}$ scales with the number of instrumented routers. With more routers, we can increase $N_{DIBS}$ to improve noise tolerance, although the rate of improvement drops off rapidly as $N_{DIBS}$ increases, making values beyond 15 unnecessary.

Similarly, smaller values for $\Delta t_{DIBS}$ will give a very inaccurate view of events, since alerts on fast scanning IP addresses will be frequently re-issued, and slower scanning worms will not be detected at all. Higher values of $\Delta t_{DIBS}$, however, place a serious performance penalty on the analysis system since all packets need to be stored for a longer period of time. Initial experimentation has shown that $\Delta t_{DIBS} > 300 seconds$ gives the best results, as values below 300 lead to too many duplicate alerts and hence noise in the scan alert data. Further experiments are needed to examine memory and CPU usage as $\Delta t_{DIBS}$ and the number of incoming ICMP-T3 messages per second increases, and to determine the best strategy for dividing the analysis across multiple processors.

For the experiments in Figure 8, $N_{DIBS}$ was set to 5, and $\Delta t_{DIBS}$ was set to 7200 and 3600 for the Code Red and Slammer run respectively. The smaller value for Slammer, taking into account Slammer's faster scan rate, makes the Slammer experiments run faster on our memory-limited hardware, but does not affect the detection results. Given the propagation speed of Slammer and Code Red v2, 3600 and 7200 are significantly larger than intuitively necessary, but the primary purpose of these values is prevent the generation of multiple alerts for the same machine and port combination within too short a time period. These values do not depend on the worm propagation speed, except that they must be significantly larger than the time it takes the worm to find and infect a single vulnerable machine. With values up to 7200, the current system can not detect *slow-scanning* worms that take days or weeks, rather than hours or minutes, to propagate.

DIBS is only half of the system, and the worm detection itself occurs in TRAFEN. In our current prototype, the process model is quite simple, but has given good results. In particular, if two scan alerts (for the same target port) occur within ten seconds of each other, TRAFEN assigns a time likelihood of 1.0 that the two scans are related; if the two scan alerts are more than 300 seconds apart, TRAFEN assigns a time likelihood of 0.0; and if the two scan alerts are between 10.0 and 300.0 seconds apart, TRAFEN assigns a time likelihood scaled linearly between 1.0 and 0.0. The time likelihood is then combined with a port likelihood of 0.9 for the same target port and 0.0 for different target ports, with the port likelihood weighted at three times the time likelihood. The same rules are used for both Slammer and Code Red v2, and are generally applicable worms propagating within hours or days. This range is not appropriate for all noise levels or slower scanning speeds, however, an issue that must be addressed with future work. Overall, the three rules capture the fact that a propagating worm generates more and more scan alerts as it infects more machines, while also capturing the fact that two scans on the same port are not necessarily related.

With the simulation and DIBS parameters and TRAFEN ruleset above, we obtain the results in Figure 8. In the case of Code Red v2, when there were instrumented routers covering two Class B networks, DIBS/TRAFEN detected the worm before it had infected 0.2% of the vulnerable machines. The infection percentage rapidly dropped to 0.03% for a coverage of sixteen Class B networks, and then remained roughly steady as the number of Class B networks increased further. Similarly, for Sapphire/Slammer, DIBS/TRAFEN detected the worm at an infection percentage of 0.01% for four Class B networks, and an infection percentage of 0.005% for sixteen Class B networks. An important difference is that DIBS/TRAFEN did not detect the Sapphire/Slammer worm at all when the coverage was only two Class B networks. Since Sapphire/Slammer propagates so quickly, and routers are configured to send only a few ICMP-T3 messages per second, DIBS/TRAFEN simply does not receive enough ICMP-T3 messages when there are only two instrumented routers.

For Code Red v2, we see that the detection results are the same when background noise is present. In addition, there were *no* false positives during a simulated 16-hour period before the worm was launched. Although the signal-to-noise ratio in the collected ICMP-T3 messages is high, DIB:S must see several probes from the same source address before generating an alert. This happens only rarely with the noise level observed at TRIUMF Canada, and thus the signal-to-noise ratio in the DIB:S *alerts* is good enough to prevent false positives. Background noise remains a critical concern, however, since we can expect that DIBS/TRAFEN will encounter more background noise in production use than what was observed at TRIUMF Canada. In addition, the background noise in the simulation was distributed uniformly across the space of source IP addresses, the opposite of what would be observed if a small group of attackers was conducting scans from a small set of source machines.

Fortunately, more complex models can address the noise problem, allow detection of slower-scanning worms, and allow faster detection of worms that bias their random scanning toward local addresses. We currently are evaluating the detection capabilities and scalability of approaches in which linear or exponential growth curves are matched against the scan alerts, as well as of Hidden Markov Models [19]. The advantage of the simulation framework is that each new approach can be evaluated against an Internet-sized address space without doing a full, real-world deployment, and the advantage of the TRAFEN framework is that each new approach is simply a drop-in replacement for the current ruleset, allowing extremely rapid development.

Overall, the results demonstrate the significant promise of worm-detection systems. The system can detect worms early in their lifetime; the system scales well, since the ICMP-

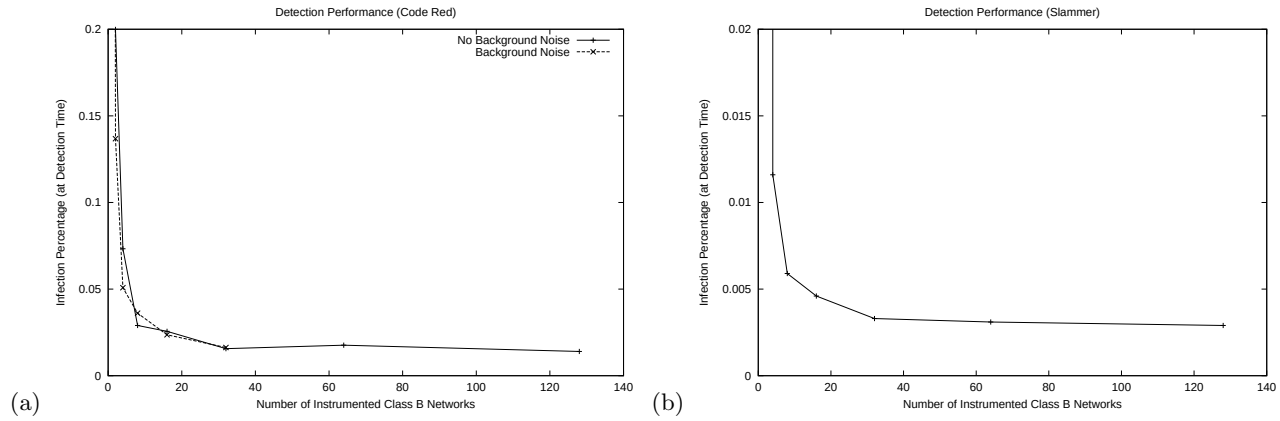


**Figure 8: The percentage of vulnerable machines infected at detection time for *(a)* a simulated Code Red v2 worm and *(b)* a simulated Sapphire/Slammer worm. Each data point is an average of three simulation runs. Two Class B networks corresponds to an address-space coverage of $2^{17}/2^{32}$, while 128 Class B networks corresponds to an address-space coverage of $2^{23}/2^{32}$.**

T3 messages are a manageable data stream be federated, each copy handling different address or port ranges; and, perhaps most importantly, researchers are beginning to work on automated response mechanisms [17] that could slow or stop even Slammer-like worms after detection.

## 7. RELATED WORK

*Worm Modeling.* Kephart and White [13] attempted to model viruses spread by floppy disks using epidemic models in the early nineties, but found that the continuous time deterministic *simple epidemic model* [10] predicted much faster virus spread than observed. They hypothesized that this was because people only exchange disks with other people they know, while the model assumes equal probability of interaction between all individuals (the *homogeneous mixing assumption*). Staniford [21] appears to have been the first to propose modeling the Code Red worm using a model that was essentially the same as the "simple epidemic model", and, in this case, it works well since random uniform scanning corresponds well with the homogeneous mixing assumption. It has since been used in several studies [22, 16, 14, 26]. However, the literature on epidemic models (e.g., [10]) includes many other variants, including stochastic formulations, discrete time models, models that include removals from the infected population—in the context of worms, this would correspond to patching, reboots, or scan filtering—and models of vector-borne diseases. The *general epidemic model*, which adds removals, has been considered in [14, 26, 17] using the continuous time formulation, while [6] considers a discrete time model.

Our work has most in common with [17, 6, 27], where models are used to study the feasibility of detection/defense system designs, but we go further since we test a working prototype system. The macroscopic level of our mixed abstraction level simulation model appears to share some common traits with the model used in [17], in terms of epidemics and topology considered, but leveraging off the SSFNet simulator we also provide packet-level capabilities.

*Worm Detection Systems.* There has been some recent work on the detection of Internet worms. The distributed NetBait system [7] does not provide automated detection of previously unknown worms, since it relies on the availability of signatures for extracting probe data from available logfiles. After the development of a signature through some other means, however, administrators can use NetBait to identify infected machines, and analyze the nature and extent of the epidemic.

Zou et al. have developed a worm-detection approach based on Kalman filters [27]. The system collects scan alerts from distributed ingress and egress monitors, and applies a Kalman filter to the scan alerts to see if the pattern of scanning activity matches their SIR-based model of worm propagation. For an address space with $2^{32}$ addresses (i.e., the Internet), monitoring coverage of $2^{17}/2^{32}$, and 500,000 vulnerable machines, their system can detect a simulated Code Red worm, and predict its overall infection rate, as soon as the worm infects approximately 5% of the vulnerable machines. Our ability to detect a Code Red infection at a lower percentage arises from our use of ICMP-T3 Unreachable messages from multiple instrumented routers, which allows us to detect even the scanning activity that hits any particular network only once, rather than waiting until the scanning activity has hit a single network enough times to be considered significant. The ingress and egress filters, or other distributed intrusion-detection systems, could provide DIBS/TRAFEN with significant additional data, however. In addition, the Kalman filter, which has several attractive features, could improve the performance of our current ruleset.

Moore, Voelker and Savage [18] use the same underlying router behavior on which we rely, and collect ICMP-T3 messages and other data to detect the "backscatter" from denial-of-service attacks. Although their system is not directly applicable to worm detection, it does illustrate that ICMP-T3 messages can be collected and used for many purposes, amortizing the cost of collecting those messages in the first place. Yagneswaran, Barford and Ullrich analyze characteristics of worm and non-worm traffic, many of which we already use, and others that might influence the development of future detection models [25]. Yagneswaran also considers passive monitoring of unused blocks of address space,

while our system can monitor both used and unused address blocks with appropriate router placements.

## 8. CONCLUSIONS AND FUTURE WORK

We have described a simulation model that combines coarse and fine grained elements to model detailed effects of large-scale worm attacks, and we have shown its usefulness in generating test traffic for the DIB:S/TRAFEN detection system prototype. We used a spatially distributed stochastic version of the continuous time general epidemic model and found it useful for modeling the early stages of the Code Red v2 and Sapphire/Slammer worms' spread.

Our experiments with the Code Red and Slammer models indicate that even the relatively simple algorithms currently implemented in the DIBS/TRAFEN system could have *i)* detected the Code Red worm before it had infected 0.2% of the vulnerable hosts while monitoring only two Class B networks, and *ii)* detected the Slammer worm at 0.01% infection while monitoring at least four class B networks. (In both cases detection was further improved with increased coverage.) These results are quite encouraging and work is continuing on the algorithms to reduce the risk of false positives in the face of background scanning noise (although this did not present a problem in a test that included a background noise level observed just before the launch of the Code Red v2 worm).

Future work includes taking advantage of the distributed execution capabilities of SSFNet to add details and scale to the worm model, and to conduct more experiments with hypothetical worm scenarios.


### ACKNOWLEDGMENTS

We thank Ken Eichman at the Chemical Abstract Service and Andrew Daviel at TRIUMF Canada for generously providing the Code Red and Sapphire/Slammer data sets, respectively.

This research is supported in part by DARPA Contracts N66001-96-C-8530, F30602-00-2-0585, NSF Grants ANI-98 08964, EIA-98-02068, CISE-0209144, and Dept. of Justice contract 2000-CX-K001. Points of view in this document are those of the authors and do not necessarily represent the official position of the United States Department of Justice.


## 9. REFERENCES


[1] Labrea. http://www.hackbusters.net/LaBrea.

[2] Ssfnet web site. http://www.ssfnet.org/.

[3] F. Baker. Rfc 1812: Requirements for IP version 4 routers. Request for Comments 1812, June 1995.

[4] Vincent Berk, Wayne Chung, Valentino Crespi, George Cybenko, Robert Gray, Diego Hernando, Guofei Jiang, Han Li, and Yong Sheng. Process Query Systems for Surveillance and Awareness. In *Proceedings of the SCI 2003*, Orlando, Florida, July 2003.

[5] Vincent H. Berk, Robert S. Gray, and George Bakos. Using Sensor Networks and Data Fusion for Early Detection of Active Worms. In *Proceedings of AeroSense 2003: SPIE's 17th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Controls*, Orlando, Florida, April 2003.

[6] Z. Chen, L. Gao, and K. Kwiat. Modeling the Spread of Active Worms. INFOCOM 2003, 2003.

[7] Brent N. Chun, Jason Lee, and Hakim Weatherspoon. Netbait: A distributed worm detection service. Available at http://netbait.plain-lab.org/., 2003.

[8] Cisco. Dealing with mallocfail and high CPU utilization resulting from the "Code Red" worm. http://www.cisco.com/warp/public/-63/ts_codred_worm.shtml, October 2001.

[9] J. Cowie, D. Nicol, and A. Ogielski. Modeling the Global Internet. *IEEE Computing in Science and Engineering*, 1(1):42–50, Jan.-Feb. 1999.

[10] D.J. Daley and J. Gani. *Epidemic Modelling: An Introduction*. Cambridge University Press, Cambridge, UK, 1999.

[11] Silicon Defense. Countermalice—Worm Containment System. http://www.silicondefense.com/products/countermalice/, 2003.

[12] S. Floyd and V. Paxson. Difficulties in Simulating the Internet. *IEEE/ACM Transactions on Networking*, 9(4):392–403, August 2001.

[13] J.O. Kephart and S.R. White. Measuring and Modeling Computer Virus Prevalence. Proceedings of the 1993 IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, May 1993.

[14] M. Liljenstam, Y. Yuan, B.J. Premore, and D. Nicol. A Mixed Abstraction Level Model of Large-Scale Internet Worm Infestations. in Proc. of the Tenth IEEE/ACM Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), Fort Worth, TX, Oct 2002. IEEE Computer Society Press.

[15] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the Slammer Worm. *IEEE Security and Privacy*, 1(4):33–39, July 2003.

[16] D. Moore, C. Shannon, and K. Claffy. Code-Red: A case study on the spread and victims of an Internet worm. in Proc. of the Internet Measurement Workshop (IMW), Marseille, France, Nov 2002. ACM Press.

[17] David Moore, Colleen Shannon, Geoffrey M. Voelker, and Stefan Savage. Internet quarantine: Requirements for containing self-propagating code. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, April 2003.

[18] David Moore, Geoffrey M. Voelker, and Stefan Savage. Inferring Internet Denial-of-Service activity. In *Proceedings of the 10th USENIX Security Symposium (USENIX'01)*, Washington, DC, August 2001.

[19] Lawrence R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceeding of the IEEE*, 77, Num. 2:257–286, 1989.

[20] Donald B. Reid. An algorithm for Tracking Multiple Targets. *IEEE Transactions on Automatic Control*, AC-24, Num. 6:843–854, 1979.

[21] S. Staniford. Code Red Analysis Pages: July infestation analysis. http://www.silicondefense.com/cr/july.html, 2001.

[22] S. Staniford, V. Paxson, and N. Weaver. How to Own the Internet in Your Spare Time. in Proc. of the USENIX Security Symposium, 2002. http://www.icir.org/vern/papers/cdc-usenix-sec02/index.html.

[23] A. Turner and M. Bing. `tcpreplay` project page (sourceforge). http://tcpreplay.sourceforge.net/, 2003.

[24] I. van Beijnum. *BGP*. O'Reilly & Associates, Sebastopol, CA, 2002.

[25] Vinod Yagneswaran, Paul Barford, and Johannes Ullrich. Internet intrusions: Global characteristics and prevalence. In *Proceedings of the International Conference on Measurements and Modeling of Computer Systems (SIGMETRICS 2003)*, San Diego, California, June 2003.

[26] C. Zou, L. Gao, W. Gong, and D. Towsley. Code Red Worm Propagation Modeling and Analysis. 9th ACM Conference on Computer and Communication Security (CCS), Washington DC, Nov 2002.

[27] Cliff C. Zou, Lixin Gao, Weibo Gong, and Don Towsley. Monitoring and early warning for internet worms. Technical Report TR-CSE-03-01, University of Massachusetts at Amherst, 2003.