Worm Detection, Early Warning and Response Based on Local Victim Information

Guofei Gu, Monirul Sharif, Xinzhou Qin, David Dagon, Wenke Lee and George Riley
Georgia Institute of Technology, Atlanta, GA 30332
{guofei, msharif, xinzhou, dagon, wenke}@cc.gatech.edu, riley@ece.gatech.edu

Abstract

Worm detection systems have traditionally focused on global strategies. In the absence of a global worm detection system, we examine the effectiveness of local worm detection and response strategies. This paper makes three contributions: (1) We propose a simple two-phase local worm victim detection algorithm, DSC (Destination-Source Correlation), based on worm behavior in terms of both infection pattern and scanning pattern. DSC can detect zero-day scanning worms with a high detection rate and very low false positive rate. (2) We demonstrate the effectiveness of early worm warning based on local victim information. For example, warning occurs with 0.19% infection of all vulnerable hosts on Internet when using a /12 monitored network. (3) Based on local victim information, we investigate and evaluate the effectiveness of an automatic real-time local response in terms of slowing down the global Internet worms propagation. (2) and (3) are general results, not specific to certain detection algorithm like DSC. We demonstrate (2) and (3) with both analytical models and packet-level network simulator experiments.

1. Introduction

In recent years, fast spreading worms have presented a major threat to the security of the Internet. Worm detection and response received renewed focus in both academia and industry. In this paper, we mainly focus on scanning worms which propagate via scanning for possible victims.

There are three characteristics common to most worms. First, many worms generate a substantial volume of identical or similar traffic. This provides the possibility of detecting known worms using their signatures (so-called misuse detection). However this approach is only effective when worm signatures are known, and fails to detect zero-day and polymorphic worms.

Second, many worms use random scanning to probe for vulnerable hosts. These scans often reach inactive IP addresses. By observing abnormally quick increases in scans to inactive IP addresses in a monitored network, one may detect the appearance of worms. Some current approaches [22, 20] use this observation. These approaches focus on global strategies and require a large monitored network (say, 2^{20} nodes) to distinguish worms from other scan activities, e.g. DDoS (Distributed Denial of Service) and port scanning. The value of these "global" worm early warning approaches is clear; however, local networks find it more useful to know which machines are infected, and how the attack is progressing. Worm detection techniques for smaller local networks should be further explored.

Random scanning worms also cause high rates of failed connections. So by observing abnormal failed connection ratios in a local network, one can detect worms. Some very recent research [18, 6] exploits this observation. However, these methods can hardly tell the difference between a worm and scanners which also cause high failed connection ratios. Further, detection based on failed connections will not detect topological worms and flash worms that use lists of victim addresses. We describe the worm detection techniques above as "symptom"-based algorithms, since they depend on artifacts or "symptoms" of worm infections, i.e., based on detecting the scanning activity associated with scanning worms.

A third factor in most worm infections is obvious, but often overlooked: vulnerable hosts exhibit infection-like behavior when infected. That is, the host is first scanned, then sends out scans destined for the same port. This is not a novel observation. Some early work in intrusion detection area [15, 16] advocated the detection of intrusion by tracing connection paths, somewhat similar to the infection relation.

We propose a simple yet powerful two-phase local worm detection algorithm, DSC (Destination-Source Correlation), for local networks. Compared with existing approaches, DSC is another step in the evolution of worm detection algorithms. Instead of only focusing on scanning symptoms, DSC aims to identify the worm victim behavior, based on both scanning pattern and infection pattern so that it can overcome some disadvantages of existing worm detection techniques. In the first phase of DSC, we correlate both incoming and outgo-

ing packets to find infection patterns. Then in the second phase we check for anomalous scanning patterns typical of worms. Thus, DSC is the first completely *behavior*-based model to detect worms. Our trace-based analysis shows that DSC is very effective and has very low false positive rates.

Our second contribution is to demonstrate the effectiveness of worm early warning using local victim information. We study how fast we can spot the appearance of new, unknown worms using a general local detection algorithm, compared to traditional "global" detection techniques. We use both an analytical model and a packet-level network simulation experiments. We exploit a new simple discrete time-based model adapted from the AAWP (Analytical Active Worm Propagation [4]) model to analyze the worm early warning performance using various scanning methods. Our results show that warnings occur when only 0.19% of all vulnerable hosts on Internet are infected, when using a /12 monitored network.

The third contribution of our paper investigates and evaluates worm response based on local victim information. Detecting worm behavior not only lets us issue alerts, but also immediately lets administrators know which victims were infected, the propagation ports, the rate of infection, and other critical information that shapes an accurate response. Using this accurate information, we propose a "local response" technique to contain the propagation of worms. Using both an analytical model and a network simulation experiment, we study the effectiveness (in terms of slowing down the global Internet worms propagation) of network level and host level response to different scanning worms. Our result shows with only 80% deployment ratio of network level local response, the Internet worms (using randomly scanning method) can be slowed down about five times than without local response. When combined with patching, the worm propagation can be slowed down at first and then stopped completely.

In the following sections we will introduce some related work and then address our three contributions separately.

2. Related Work

In studying malware propagation, Kephart *et al*. [7] proposed a classic epidemiological model to describe virus/worm spread. Zou *et al*. [23] proposed a modified "two-factor" epidemic model that considered the situation of human countermeasures and congestion due to worm traffic. In [4], Chen *et al*. proposed a discrete time-based propagation model to track the spread of worms, considering the effects of patching and worm cleaning during the worm propagation.

Multiple approaches have been proposed for worm early detection. Staniford *et al*. [14] first proposed an idea of establishing a cyber "Center for Disease Control". In collecting information of worm activities, Moore [9] proposed the use of distributed "network telescopes" using a reasonable large fraction of the IP space to observe security events occurring on the Internet. Using honeynets to gather and identify attacks was also proposed and implemented [11, 5]. In [15, 16], researchers proposed the detection of intrusion by tracing connection paths through the departments of an organization. This is somewhat similar to our DSC approach. However, DSC is more simple and practical, dedicated to worm detection, and combines both infection pattern and scanning pattern anomaly detection techniques.

In the area of worm early detection/warning, Zou *et al*. [22] proposed a Kalman filter-based detection algorithm. This approach detects the trend of illegitimate scans to a large unused IP space. In [20], Wu *et al*. proposed a victim counter-based detection algorithm that tracks the increased rate of new infected outside hosts. Recently, Jung [6], Weaver *et al*. [18] proposed worm containment based on the observation that scanning worms cause high failed connection ratios. Our proposed detection algorithm is a different approach combining the infection nature of worm and an anomaly scan detection technique. It effectively detects all kinds of fast spreading scanning worms, including topological worms.

In the area of worm response/defence, Moore [10] only focused on a more basic question: How effective can any ("global") containment approach counter a worm on the Internet? Zou [24] proposed using a dynamic quarantine defense on the principle "assume guilty before proven innocent". Williamson [19] proposed the idea of rate limiting on the host by limiting the number of new outgoing connections. Our "local response" is a different approach based on local victim information which combines accurate, early detection and systematic immediate, automatic, real-time response.

3. Local Victim Detection Using DSC

There are two basic requirements for any worm victim detection algorithm to detect worms on local networks: speed and accuracy. Local network detection systems should detect victims as early as possible. False positive rates must also be kept to a minimum.

In this section we will propose DSC (Destination-Source Correlation) as a candidate local victim detection algorithm. As noted above, current worm detection approaches mainly focus on scanning patterns. We believe the infection *behavior* is a very important common feature of worms. Therefore, we designed our worm victim detection algorithm by first considering the worm infection pattern.

Worm infections are often different, but in a general sense, many follow a common pattern. After a vulnerable host is infected by a worm on a port i (i.e., the host is the destination of an early worm attack), the in-

fected host will send out scans to other hosts targeting at the same port i in a short time (i.e., the infected host is a source of new worm attacks). For example, an infected host by Slammer worm on port 1434 also sends out scans to port 1434 on other hosts. Our detection model tracks hosts who exhibit this "infection behavior", and identifies this destination-source infection pattern.

3.1. Basic Idea of DSC Algorithm

The DSC algorithm includes two phases. First, we find the infection-like pattern. Second, we check the abnormal outgoing scan rate for suspicious hosts observed in first phase. In section 5, we will discuss "local response", which can be considered as the third phase when we automatically quarantine the victim's outgoing traffic at the right port.

Assumption: Clearly, DSC does not aim to detect all types of worms. It is unlikely any one algorithm could detect all manner of malware. Instead, DSC aims to detect scan-based, *fast* spreading worms. Further, we presume that the infection time for hosts is not very long. In other words, DSC may not effectively detect email worms, very slow scanning worm, or sleeper worms with very slow rates of infection. Compared to the fast spreading worms like SQL slammer and CodeRed, slow spreading worms do less damage to networks (but not necessarily hosts), and are easier to contain, in part because their slower spread rate allows for human intervention.

General idea: We keep a sliding window of local network traffic (destination addresses of SYN and UDP traffic). Two general items are tracked: for each port witnessed in this traffic, we record the address of the inside destination host and the scanning source from the monitored network. The addresses can be IP, or MAC address. If a source scan originates from a host that previously received a scan on an identical port, i.e., we observe a worm behavior-like infection pattern, we then treat this local host as a suspicious victim. In other words, if a host gets a packet on port i, and then starts sending packets destined for port i, it becomes suspect. Then if the immediate outgoing scan rate for the suspect hosts deviates from a normal profile, the suspicious victim is considered to be infected. We then output a worm alert and indicate the victim IP and its scan rate. By training for a normal profile of infection-like scan rates, we can achieve a very low false positive rate in detecting worm victims. Section 3.3 discusses our anomaly detection technique. By correlating incoming and outgoing traffic plus anomaly scan detection, DSC therefore focuses on worm-like behavior, instead of just the scanning symptoms of worms.

3.2. Bloom Filter-based DSC Algorithm

Detecting infection-like behavior in hosts requires keeping state. For a large network with heavy traffic we describe a practical implementation using space-efficient Bloom filters [3]. Conceptually, a Bloom filter is a bit vector, initially all zero. As entries are made to the filter, k independent functions are used to hash data into indicators, and the corresponding bit is set to one. To see if data has been entered into a Bloom filter, on merely performs the k hash operations on the data, and verifies all corresponding bits are set to one. Thus, Bloom filters provide a space efficient way to "remember" data.

For every port, we use Bloom filter to store the destination addresses for scanning traffic (SYN and UDP) directed at the network. For scanning traffic originating form the network, we check if the source host appears in the corresponding filter. If so, we record this infection-like behavior in a watch list. If the same host repeatedly sends out more packets and exceeds a trained threshold, we issue an alert.

Because Bloom filters do not have a mechanism to reliably remove old entries, we cannot use a single Bloom filter as a sliding window. In DSC, we use two Bloom filters to simulate a sliding window for every port. Every filter keeps the information for a time periods T_b. (In theory it is one time tick, but to detect worms with a longer infection time, a bloom filter might record traffic longer than just one time tick.) Suppose filter1 is in use at time 0 and we begin to insert addresses into it. After $T_b/2$, we will insert addresses to both filters. After T_b, filter1 is reset to zero and we begin to use filter2. By switching two Bloom filters periodically, we roughly simulate a sliding window.

When using a Bloom filter, the potential false positive rate can be calculated as $FP \leq (1-(1-\frac{1}{m})^{kn})^k \approx (1-e^{-\frac{kn}{m}})^k$. Here m is the size of the filter in bits, n is the total number of hosts inside the monitored network, k is the number of hash functions.

3.3. Detection of Abnormal Scan Rate

Merely identifying the basic infection pattern in traffic is not enough to mark hosts as truly infected by a worm. Indeed, there are some infection-like patterns in legitimate traffic, such as telnet/ssh chains and P2P connections. To distinguish legitimate from infectious traffic, we further consider the high rate of outgoing scanning that often accompanies worms. Existing scan detection algorithms like TRW [6] can be applied here. In this paper, we just simply use anomaly detection heuristics to identify this unusual pattern. In practice, we first establish the normal profile of the outbound scan rate of services which exhibit infection-like behavior. In idle networks, or networks that exhibit no fast infection-like behavior during a training period, one can merely pick

an arbitrarily reasonable threshold. For example, if scan rates of infection-like services are quite low (e.g. less than 3 /sec), then a higher number of such events (e.g. 6 /sec) are all that is needed to issue an alert.

For networks with various infection-like (but normal) behavior, the simple detection heuristic discussed above may not work. We use Chebyshev's inequality to determine whether the simple detection heuristic can be used.

For a given random variable, Chebyshev's inequality can provide an upper bound on the probability that the value lies outside a certain distance from the variable's mean. During the training phase, we approximate the mean μ and the variance σ^2 of the real scan rate distribution to each port by computing the sample mean μ and the sample variance σ^2 for the scan rate $s_1, s_2, \ldots, s_n$. The intuition of applying Chebyshev inequality is that we can set a threshold t so that we get a bound on the probability that a value (denoted as x) deviates from the mean μ more than the threshold t. In particular, a Chebyshev inequality can be represented as $p(|x - \mu| > t) < \frac{\sigma^2}{t^2}$, where p is probability.

Based on the inequality, when σ^2 is large, then in order to have a reasonable bound, we need to select a large t. But this may let the worm detector admit high scan rate behavior, resulting false negative in worm victim detection. Thus, we apply our simple detection heuristic only when the values of μ and σ^2 are small.

In fact, with real-world data (see our following experiments), most normal infection-like traffic has very low immediate scan rates, e.g. less than 3 /sec. So we can set our detection threshold to be a reasonable value, say, 6 /sec.

3.4. DSC in Practice

To test DSC, we used trace files collected from two distinct sources. Since we know no worm appears in the traffic traces, the main purpose of the experiments is to evaluate the *false positive rate* of DSC.

First, we used data from the Waikato Internet monitoring project (WAND) [17] to train DSC. We selected a 65GB (compressed) trace sample, representing a continuous six and one half week trace between February and April 2001 at the University of Auckland uplink.

We split the data into two chronological parts for training and testing respectively, in particular, the first 80% of the trace was used as training data and the rest 20% is used for testing the false positive rate. This simulates the process of learning from the historical data and applying the learned model to current and future data. We first established a normal profile of scan rate for every port immediately after infection-like behavior. Thus, if a host received traffic on port i, and then generated traffic to remote port i, we measured the outgoing rate. We focused on traffic from some representative TCP ports, i.e., 21, 22, 23, 25, 80, 139, 445 and UDP ports, i.e., 53, 1434.

Surprisingly we did not find any *infection-like behaviors* on all selected ports except port 80. There were 25 infection-like behaviors total on port 80, however, the immediate outgoing scan rate after infection-like behavior to external port 80 proved to be very small, i.e., $\mu = 0.1, \sigma^2 = 0$. So we set the detection threshold to be 2 scans/sec. Testing results showed that the false positive rate is 0% in this experiment.

Network use of course varies widely. So, we contrast the WAND results with monitoring traces taken from the Georgia Institute of Technology, College of Computing (GTTrace). The GTTrace data came from a 6-day capture of 2GB busy backbone.

For this second round of experiments, we also devoted the first 80% of the data for training and the rest for testing. Since P2P activities are very normal on campus networks, in addition to the commonly used ports (e.g., TCP port 22, 25, etc.), we expanded our analysis to include TCP ports used by some popular P2P applications. In particular, we monitored the traffic on TCP ports 6881 to 6889 used by BitTorrent [2], a P2P program.

Port	Infection-like behaviors #	μ	σ^2
22	19	0.1263	0.0020
23	8	0.3625	0.0512
80	451	0.7978	1.8120
6884	8	1.9375	1.0455

Table 1. GTTrace Analysis

Table 1 shows the observations and results of normal infection-like behaviors on some selected ports. From the Table 1, we can see that the number of infection-like behaviors is in fact very small in our normal live network. The corresponding average outbound scanning rate is also well below that of worms, e.g., CodeRed or Blaster. We set the detection threshold to be 2 scans/sec for port 22, 23 and 6 scans/sec for port 80, 6884. As expected, the detection system did not output worm alerts in the testing phase either.

In [12] DSC algorithm was applied to data collected from honeynets during the breakout of SQL Slammer worm. Results showed that the DSC algorithm can detect the Slammer worm at an early stage without false alarms. Further, the algorithm also determined all of three victims' IP information immediately after they are infected.

3.5. Limitations of DSC

Besides our assumptions (discussed above), there are some limitations to DSC. Some normal applications can produce infection-like traffic and may not have an immediate stable scan rate. For example, hosts run-

ning gnutella may receive TCP/6346 traffic, and also send data to other clients through TCP/6346 elsewhere. Moreover, these connections come and go in bursts, based on human interactions. Another instance may be a busy smtp relay server. In another word, these applications demonstrate a normal high μ, σ^2. From Chebyshev's inequality we can see that the simple detection heuristic cannot be used. In such a case, normal traffic blacklisting might help, but this may provide worms with an unwatched vector for transmission. In future work, we will study some other anomaly detection algorithms besides considering the rate of outgoing connections.

DSC has another limitation when it comes to bipartite, or dual worms. That is, worms that use two (or more) attack vectors can evade correlation. Let's assume a worm exploits two vulnerabilities, A and B. Attackers can code the worm so that it alternates what attack it uses after each infection. Thus, after spreading first by using attack A, it next spreads using B, and vice versa. This concern is not merely theoretical. Some malware such as Phatbot uses at least 8 different attack vectors. Multi-vector worms present a challenge for DSC.

We designed DSC to be effective for the common case of fast spreading worms, and not a perfect algorithm for all cases. Our intent was to illustrate the value of using local detection strategies. Although DSC has these limitations, we believe it offers a promising approach to detect local victims in real-time. DSC certainly is not the only approach. It does, however, suit our goal: identify a feasible local detection strategy that offers speed and accuracy.

4. Worm Early Warning Using Local Victim Information

In this section, we consider the effectiveness of local detection systems. We presume that an administrator has selected a local victim detection algorithm, one with a high detection rate and low false alarm. (It may be DSC, or some other algorithm.) We study the following problems: Is the local detection system suitable for fast early warnings? What is the effectiveness of worm early warning using local victim information? We use analytical models and network simulation experiments to answer these questions.

In this paper, we will use a discrete time-based Analytical Active Worm Propagation-like model (AAWP) as our analytical model. (For an overview of AAWP, see [4]) We differ from the traditional AAWP model on two points. First, previous AAWP models and others based on epidemic model, e.g. [23], assume vulnerable hosts are uniformly distributed in the whole IPv4 space. But in fact, not every IPv4 address is allocated. The real vulnerable hosts may be uniformly distributed in those allocated IP space, instead of within the whole IPv4 space. The size of allocated IP space is about 10^9 [21, 20]. We denote this allocated IP space as T. Second, we focus on local victims information instead of the global view. If the local network we monitor is the whole real assigned space, then this becomes the global victim information.

4.1. Uniform Scan

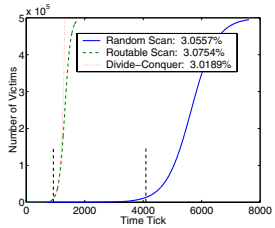
Uniform scanning means that a worm randomly selects a host in its scanning space as the scanning target. There are three types of uniform scans: random scan, routable scan and divide-conquer scan.

Random scan: This is a common scan strategy used by many worms, e.g., Code Red, Slammer. Because we assume the vulnerable hosts are uniformly distributed in the real assigned IPv4 space, we only focus on these $T = 10^9$ space (all victims are located in this space). Ω is the whole IPv4 space ($\Omega = 2^{32}$), N is the total number of vulnerable hosts on Internet, s is the scan rate (per time tick), n_i is the number of infected hosts at time tick i. The scans entering space T at time tick $i+1$ should be $k_{i+1} = sn_i\frac{T}{\Omega}$. Within this space T, the chance of one host being hit is $1-(1-\frac{1}{T})^{k_{i+1}}$. Then we have new worm propagation equation Eq. (1).

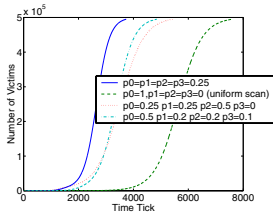
$$n_{i+1} = n_i + [N - n_i]\left(1 - (1 - \frac{1}{T})^{sn_i\frac{T}{\Omega}}\right) \qquad (1)$$

If we monitor a network with size D, then the number of local victims at time tick i is just $v_i = \frac{D}{T}n_i$ because we assume that victims are uniformly distributed in the assigned IPv4 space T.

Because of space limitations, we omit an analysis of routable scans and divide-conquer scans. Their propagation formulas are very similar to that of random scan. For more detail, please refer to [12]. Figure 1(a) shows these three worm propagations. In the paper we will fix N=500,000 and Hitlist (initial number of worm)=1.



(a) Three Uniform Scans



(b) Local Preference Scan and Sequential Scan are faster than uniform scan

Figure 1. Uniform and Local Preference Scan (scan rate=20/time tick)

4.2. Local Preference Scan

Some worms (e.g. Code Red II, Nimda) have scanning policies with a local subnet preference. Without loss of generality, a local preference scan policy can be summarized as the following: with p_0 probability, select a totally random address; with p_1 probability select a random address with the same first octet (same class A address); with p_2 probability select a random address with the same first two octets (same class B address); with p_3 probability select a random address with the same first three octets (same class C address). Here $p_0+p_1+p_2+p_3=1$. For Code Red II, $p_0=1/8, p_1=1/2, p_2=3/8, p_3=0$. For Nimda, $p_0=1/4, p_1=1/4, p_2=1/2, p_3=0$. If $p_0=1, p_1=p_2=p_3=0$, this becomes a uniform scan worm discussed above.

To analyze the local preferences of a local scanning worm, we divide all the assigned IPv4 space into $b=T/2^8$ small pieces so every piece is a /24 subnet. We then divide all these b subnets into four different types. First, we have a special subnet (denoted by Subnet type 1) which has a histlist of h_1. In this context, "special" means this subnet has a big histlist (most or all of the first worms are here). Second, we identify (2^8-1) subnets having the same first two octets as the first subnet, but exclusing Subnet type 1. We denoted this subnet as Subnet type 2, with a hitlist of h_2. Third, we identify $(2^{16}-2^8)$ subnets having the same first octet as the first subnet, excluding both Subnet type 1 and 2. We denote this subnet as Subnet type 3, with a hitlist of h_3. Finally, we identify $(b-2^{16})$ subnets having different first octet as the first subnet, denoted by Subnet type 4, with a hitlist of h_4.

Within a time tick, by summing the average number of scans coming from a local subnet and other different types of subnets, the average number of scans in these four kinds of networks is:

$$\begin{aligned} k_1 &= p_3 s V_1 + p_2 s g_1/2^8 + p_1 s g_2/2^{16} + p_0 s g_3/b \\ k_2 &= p_3 s V_2 + p_2 s g_1/2^8 + p_1 s g_2/2^{16} + p_0 s g_3/b \\ k_3 &= p_3 s V_3 + p_2 s v_3 + p_1 s g_2/2^{16} + p_0 s g_3/b \\ k_4 &= p_3 s V_4 + p_2 s v_4 b/2^{24} + p_1 s v_4 b/2^{24} + p_0 s g_3/b \end{aligned} \quad (2)$$

where $g_1=V_1+(2^8-1)V_2$, $g_2=V_1+(2^8-1)V_2+(2^{16}-2^8)V_3$, $g_3=V_1+(2^8-1)V_2+(2^{16}-2^8)V_3+(b-2^{16})V_4$.

Here V_i is the victim number of subnet type i, and its initial value is h_i. k_i is the scan number in subnet type i.

In every subnet, the number of vulnerable machines is $2^8(N/T)$, and T is the real assigned IP space. So the local victims at time tick $i+1$ can be calculated as

$$v_{i+1}=v_i+(2^8N/T-v_i)[1-(1-1/2^8)^{k_i}]$$

For a randomly selected /24 network, the average number of victims can be calculated as $V_1/b+V_2(2^8-1)/b+V_3(2^{16}-2^8)/b+V_4(b-2^{16})/b$. Suppose we monitor some random networks and the whole size is a $/x$ network. Then we have the average number of victims v_e:

$$\begin{aligned} v_e = \; & \frac{2^{32-x}}{2^8}(V_1/b+V_2(2^8-1)/b \\ & +V_3(2^{16}-2^8)/b+V_4(b-2^{16})/b) \end{aligned} \quad (3)$$

When $\frac{2^{32-x}}{2^8}=b$, then it shows all the victims on the Internet.

Figure 1(b) shows that different local preference policies will affect the rate of spread. Chen *et al.*[4] argue that the spread of local preference scanning is somewhat similar to that of random scanning, in fact a little "worse". However, in our analysis (Figure 1(b)) we surprisingly find that in fact it is much faster than a random scan worm. This is true even when we use the same parameter scan policy and hitlist discussed in [4]. We believe the reason is that Chen *et al.* assume that the real vulnerable hosts are uniformly distributed in the whole IPv4 space, but in fact they are not. The vulnerable hosts are only uniformly distributed in the real *assigned* IPv4 space. That's about $1/4$ of whole IPv4 space. This assumption artificially lowered the number of vulnerable hosts in the network. In turn, this caused the worm's spreading speed to be slower than reality. Our model, however, is more accurate and can show the real speed of a local preference scan worm.

4.3. Sequential Scan

Sequential scanning means after randomly selecting a starting IP y, a worm will continue to scan $y+1$ (or $y-1$), and so on. Blaster is a typical sequential scan worm. Without loss of generality, the policy of sequential scan is combined with local preference. This means worm can have four probabilities to choose the start point at different kinds of networks. Thus we use the same $p_0, p_1, p_2, p_3, k_1, k_2, k_3, k_4$ discussed above. But computing the number of victims in time tick i is a little different because for sequential scans the chance of hitting one host in subnet type i is $k_i/2^8$.

$$v_{i+1}=v_i+(2^8N/T-v_i)k_i/2^8$$

From this equation we can see that the propagation of sequential scans is very similar to the local preference scan.

When randomly monitoring a $/x$-sized network, the average number of victims can be computed using the same Eq.(3).

4.4. Worm Early Warning Performance with Various Scan Methods: Analytical Results

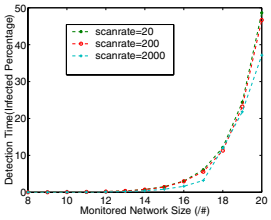
With the analytical model for various scan techniques discussed above, we evaluate the performance of early warning systems that use local victim detection systems (e.g., DSC discussed above). We evaluate the detection

time (Table 2) in terms of infected percentage of the whole Internet's vulnerable hosts when at least one infected victim in our monitored network is identified, i.e., the time when $v_i \geq 1$. This is a reasonable assumption, considering the very low false positive of our DSC algorithm. And we will also discuss the situation when only one victim reported by some victim detection algorithm is not sufficient to issue an alert.

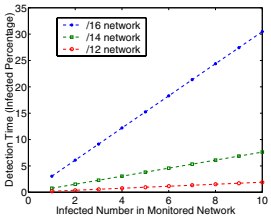
Scan Type	DT (/12 Vs /16)	IP(/12 Vs /16)
Random	2953/4157	0.1913%/3.0557%
Routable	691/973	0.1914%/3.0754%
Divide-Conquer	691/1711	0.1918%/3.0530%
Local Preference	1071/1711	0.1913%/3.0610%
Sequential	1073/1715	0.1908%/3.0582%

Table 2. Analytical Early Warning Time

Table 2 shows detailed information about the worm detection time and corresponding infection percentage of different scan methods using /12 and /16 networks respectively. Here DT denotes detection time in time tick, IP denotes infection percentage. The scan policy in local preference and sequential is $(p_1 = p_2 = p_3 = p4 = 1/4)$. Scan rate=20 per time tick. It is easy to see that we can effectively detect *all* the scan methods discussed above at almost the *same early* stage, i.e., less than 0.19% when using /12 networks, and less than 3.08% infection percentage even using a /16 small local network. If we use a larger network (maybe the union of some small networks), the detection performance will surely be better.



(a) Effect of different scan rates: routable scan



(b) Linear Relationship. Scan rate=20 scans/time tick

Figure 2. Detection Performance

Figure 2(a) shows the detection performance using different monitored network sizes. It is very clear that as the size of the monitored network increases, detection performance improves. It also shows that using different scan rates has almost no effect on the detection time. Worm detection using local victim information-based algorithms is very stable and not sensitive to both different scan methods and different scan rates.

There might be some concern about the validity of an alert issued when there is only one victim detected. Although a local detection algorithm such as DSC discussed above has a very low false positive rate (nearly zero after training), in some cases it is better to find several detected victims before issuing a warning. We studied the relationship between the number of infected hosts in the monitored network and the detection time as shown in Figure 2(b). The relationship turns out to be linear. Using a /12 network, with 2 victims the warning can be issued when only 0.38% of all vulnerable hosts on Internet are infected. Without loss of generality we suppose the Bayesian detection rate (that an alarm really indicate a worm [1]) of the local victim detection algorithm is P_b, then after c worm alarms are issued, the probability that this is a real worm is $1-(1-P_b)^c$. Given the total Bayesian detection rate we wish to achieve, we can easily compute the number of alarms we need.

It should be noted that there is an inherent (unavoidable) risk of false alarms with any statistical based early warning strategy. A comprehensive worm detection and response framework needs to include dynamic feedback control mechanisms to (continuously) select the optimal response based on the current status of worm infection. This is a future work, and is orthogonal to any worm early warning work, including ours. On the whole, compared with other detection algorithms, our candidate algorithm, i.e. DSC, improves the quality of the data stream used for worm early warning, since random worm-like behavior is more suspicious than random network scans.

4.5. Network Simulator Experiments

To test the local warning system, we use a worm model developed in the packet level network simulator GTNetS (Georgia Tech Network Simulator) [13]. Because this is a real packet level simulator (which includes network router congestion, TCP latency, and other stochastic events) every run takes time. A small network ($/16$) was simulated with a realistic packet level worm model instead of using a partial analytical model for the whole Internet such as that used in SSFNET [8].

Our motivation for using packet level network simulation is to validate our local warning system and the results of our analytical models in an Internet-like setting. We used a hybrid network topology with clustering backbone and hierarchical sub-networks. The address space was populated with uniformly random hosts. Then, the vulnerable hosts were selected from the population in a uniform random way, $\Omega = 2^{16}, T = \Omega 3/4, N = 32000, Hitlist = 1$. Since the simulated network is small compared to the whole Internet, we set the time required to detect a worm as negligible. Thus, whenever a host in a monitored space became infected, it was detected and the number of infected hosts in the entire network was recorded.

We use three different monitored network sizes to see how fast we could detect random scan worms. We ran our experiment 15 times for each monitored network size and computed the average and variance. The result is shown in Table 3 (early warning time based on local victim information).

Monitor Space	2^9	2^8	2^7
Avg Scan Rate	5.44	5.50	5.31
Avg Detection Time	0.64%	0.71%	2.39%
Variance	0.362	0.484	7.922
Analytical Detection Time	0.68%	0.71%	2.29%

Table 3. Network Simulator Experiments

We can clearly see that using a smaller network, the detection time will be longer and the variance will be larger. Further, the detection time of the network simulation experiments matched the output of our analytical model when given identical input parameters.

5. Local Response

5.1. Analytical Model for Local Response

With the victim information provided by a local victim detection algorithm (e.g., DSC), we can automatically take immediate and accurate responses that block victims so as to effectively stop the outgoing propagation. The policy decision to block local victims can be accomplished entirely within the local network in real-time. This contrasts with global response strategies, i.e. Internet Quarantine [10], which require complex and time consuming coordination between CDC-like authority and Internet routers. Our local response is kind of a passive quarantine, which only throttles local victims and prevents further outgoing spreading. This is somewhat similar to Williamson's idea [19] which limits the rate on the host by limiting the number of new outgoing connections.

Local response can be more effective, since local administrators know details about the victim machines and take more accurate action to block (not rate limit) the outgoing connections of victims (not all hosts) at that port (not all ports). This local response policy can be deployed on every host as [19]. We call this approach "host level local response". But it is probably more effective to only deploy on the edge router of LAN (network level local response) so that every LAN only need one position to deploy such a quarantine policy.

Suppose there are b_1 Class A networks, b_2 Class B networks, b_3 Class C networks using network level local response policies (based on local victim information provided by some worm victim detection algorithm like DSC). Then the size of the network using local response is $D = b_1 2^{24} + b_2 2^{16} + b_3 2^8$. Let q_{i+1} denote the outgoing scans blocked by edge routers of these networks. For a random scan worm, we have

$$v' = \frac{2^{24}}{T} n_i; v'' = \frac{2^{16}}{T} n_i; v''' = \frac{2^8}{T} n_i$$

$$\begin{aligned} q_{i+1} = \quad & s \cdot [b_1 v'(1 - 2^{24}/\Omega) + b_2 v''(1 - 2^{16}/\Omega) \\ & + b_3 v'''(1 - 2^8/\Omega)] \end{aligned}$$

$$n_{i+1} = n_i + (N - n_i)(1 - (1 - 1/T)^{(sn_i - q_{i+1})\frac{T}{\Omega}})$$

We can imagine that using different values for b_1, b_2, b_3, the quarantine effect will be somewhat different. It also seems that given the same size of D, using smaller networks (e.g. more Class C network instead of more Class A network), the quarantine performance will somewhat improve. The heuristic is that in smaller networks, there's little chance for scans to be directed inside, i.e., a larger percentage of scans are outside and will be quarantined.

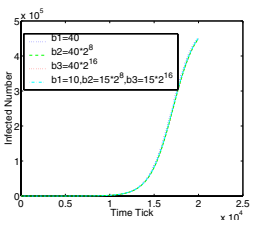
However, we find that this is not the case for random scan worms. From Figure 3(a) we can see that given the same size of D, whatever b_1, b_2, b_3 are, the quarantine performance is almost the same. This is because for a random scan worm, whatever kind of class network it is (A, B or C), the probability of producing inside scans and also hitting inside vulnerable hosts is almost negligible. Based on this observation, we can simplify the formula for local response using $\alpha = D/T$ as the only parameter to denote the deploy rate of a local response. We call α the quarantine rate, so that in every time tick, on average there will be α percent of scans blocked by local response. Now we have

$$n_{i+1} = n_i + (N - n_i)(1 - (1 - 1/T)^{sn_i \frac{T}{\Omega}(1-\alpha)})$$

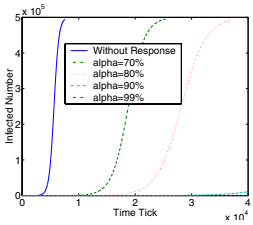
The quarantine effect of host level response to random scan worm is almost the same as network level response (and they can use the same formula). Figure 3(b) shows the effect of our local response (whatever network level or host level). With only 80% deployment ratio of network level local response, the Internet worms (using randomly scanning method) can be slowed down about five times than without local response. Using a higher deploy rate (quarantine rate) α, containment improves. When $\alpha \geq 90\%$, or when almost all networks deploy a local response policy, the worm is nearly stopped from the beginning!

Now consider a local preference scan worm. Because using network level local response does not affect the internal scans, we have the similar formula as Eq.(2) except that $k_1 = p_3 s v_1 + (1 - \alpha)(p_2 s...)$, and so do k_2, k_3, k_4.

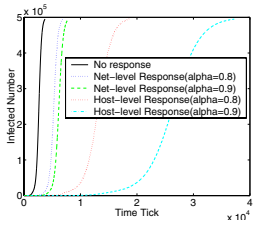
We imagine that for local preference scan worms, the effect of network level local response is worse than for uniform scanning worms. This is because network level local response can only prevent victims from infecting



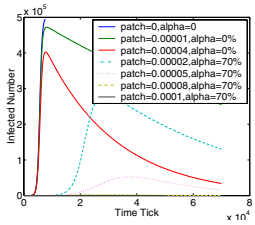
(a) Effect of different b1,b2,b3 given same size $D = 10 * 2^{24}$



(b) Effect of local response to random scan worm



(c) Effect of local response to local preference scan worm with policy $p_0 = p_1 = p_2 = p_3 = 0.25$



(d) Effect of both quarantine rate and patching rate. $t_r = 2953 + 100$. 2953 is the detection time using a /12 monitor, and 100 is additional delay.

Figure 3. Effect of Local Response (scan rate=20/time tick)

outside network hosts but cannot prevent worm propagation within local network, which is worse with local preference scans.

For local preference scan worms, a more effective approach is to use host-level local response because it can block more scans than a network-level local response. For instance, in this case, $k_1 = (1-\alpha)(p_3 s v_1 + ...)$ and so do k_2, k_3, k_4.

Figure 3(c) clearly shows how network-level and host-level responses slow down the local preference worm spreading with $p_0 = p_1 = p_2 = p_3 = 0.25$. Compared with network-level response, the performance of host-level response is better. Host level response can slow down by nearly 5 times, but network level response only slows down about 2 times.

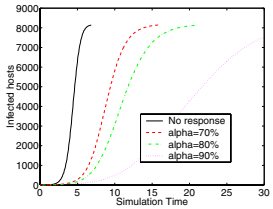
In practice, quarantine can only slow down a worm, and cannot decrease the number of victims. Because we know detailed information about local victims, we can easily take more aggressive and focused actions to immunize infected machines. So in addition to a quarantine rate we also consider the effect of a delayed patching rate p, applied after worm detection and analysis, with the response time t_r, the delay before people learn about the need to patch. When $i > t_r$,

$$n_{i+1} = (1-p)n_i + [((1-p)^{i-t_r} N - n_i) \cdot (1-(1-1/T)^{s n_i \frac{T}{\Omega}(1-\alpha)})]$$

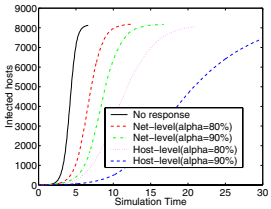
Figure 3(d) shows the effect of both patching and quarantine. We can see that if we only use patching, the worm will first infect most of the vulnerable hosts, and then slow down. So it is better to use both quarantine and patching to first slow down the spread and then stop the propagation.

5.2. Simulation of Local Response

We also use our packet-level worm simulator to model the effect of both network and host level local response in a small network (a /18). We set $\Omega = 2^{14}, T = 3/4\Omega, N = 2^{13}$ in this experiment. The host level response method was simulated by making an infected host stop sending attack packets after a specified count. For the network level response, the size of the local networks was set to be /27 in size. When a host is infected, all worm packets going out of the same /27 network were blocked, and vulnerable hosts inside could still be infected. The test was done for both uniform random and local preference scanning (scan policy: 0.25 in same /27, 0.25 in same /24, 0.25 in same /21, 0.25 in same /18). The result is shown in Figure 4.



(a) Local Response for Random Scan Worm



(b) Local Response for Local Preference Scan

Figure 4. Network Simulator Experiments

As it can be seen from the Figure 4, the local response method was able to slow down the spread of the worm. With large numbers of hosts having a response capability, the worm propagation was slower. To compensate for the randomness of the network and target selection variation, each run was done several times to create an average propagation.

Also we can clearly see that the network level response is effective for uniform scanning, but not as effective for local preference scanning as demonstrated in the analytical model.

6. Conclusion

In the paper, we introduced a systematic approach to worm detection, early warning and response based on local victim information. The advantages of our approach are the following.

First, we give an example of a local victim detection algorithm DSC (Destination-Source Correlation). By correlating incoming and outgoing traffic together with anomaly scanning detection, DSC therefore focuses on worm-like behavior, instead of just the scanning symptoms of worms. DSC can be used in production networks, and is complementary to other existing worm detection algorithms.

Second, with distributed deployment of local victim detection sensors, we can detect worms early, even without using a very large monitored network. Additionally, the alerts include details about the victims, improving the value of the warning to local administrators. And, the local victim information approach is not sensitive to worm scanning techniques and has good performance for all scan techniques.

Finally, based on local victim information we can use accurate, real-time, effective and practical local response to slow down and stop the propagation of worms. We believe this local response can be better than global response which is time-delayed, less effective and somewhat impractical for Internet routers.

We conclude that a sensitive local detection algorithm can be very helpful, both for early warnings, and for slowing (or stopping) the propagation. DSC is one promising candidate despite its limitations. We encourage others to identify and explore more powerful local detection algorithms. In the future work, we will improve DSC and incorporate dynamic factors for local responses.

7. Acknowledgments

We gratefully thank Mr. Cliff C. Zou, Zesheng Chen and Christian Kreibich for helpful suggestions and discussions on the research work. This work is supported in part by NSF grants CCR-0133629 and CCR-0208655 and Army Research Office contract DAAD19-01-1-0610. The contents of this work are solely the responsibility of the authors and do not necessarily represent the official views of NSF and the U.S. Army.

References

[1] S. Axelsson. The base-rate fallacy and its implications for the difficulty of intrusion detection. In *Proceedings of ACM CCS'1999*, November 1999.

[2] Bittorrent. http://bitconjurer.org/BitTorrent/.

[3] B. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of ACM*, 13, July 1970.

[4] Z. Chen, L. Gao, and K. Kwiat. Modeling the spread of active worms. In *Proceedings of the IEEE INFOCOM 2003*, March 2003.

[5] D. Dagon, X. Qin, G. Gu, W. Lee, J. Grizzard, J. Levine, and H. Owen. Honeystat: Localworm detection using honeypots. In *Proceedings of RAID'2004*, September 2004.

[6] J. Jung, S. E. Schechter, and A. W. Berger. Fast detection of scanning worm infections. In *Proceedings of RAID'2004*, September 2004.

[7] J. Kephart, D. Chess, and S. White. Computers and epidemiology. 1993.

[8] M. Liljenstam, D. Nicol, V. Berk, and R. Gray. Simulating realistic network worm traffic for worm warning system design and testing. In *Proceedings of ACM WORM'2003*, Oct 2003.

[9] D. Moore. Network telescopes. http://www.caida.org/outreach/presentations/2002/usenix_sec/, 2002.

[10] D. Moore, C. Shannon, G. M. Voelker, and S. Savage. Internet quarantine: Requirements for containing self-propagating code. In *Proceedings of the IEEE INFOCOM 2003*, March 2003.

[11] N. Provos. A virtual honeypot framework. In *Proceedings of 13th USENIX Security Symposium(Security'04)*, August 2004.

[12] X. Qin, D. Dagon, G. Gu, W. Lee, M. Warfield, and P. Allor. Worm detection using local networks. Technical report, College of Computing, Georgia Tech, Feb 2004.

[13] G. F. Riley, M. I. Sharif, and W. Lee. Simulating internet worms. In *Proceedings of IEEE/ACM MASCOTS'2004*, October 2004.

[14] S. Staniford, V. Parxon, and N.Weaver. How to Own the Internet in Your Spare Time. In *Proceedings of 2002 Usenix Security Symposimum*, 2002.

[15] S. Staniford-Chen, S. Cheung, and etc. GrIDS–a graph-based intrusion detection system for large networks. In *Proceedings of the 19th National Information Systems Security Conference (NISSC'96)*, October 1996.

[16] T. Toth and C. Kruegel. Connection-history based anomaly detection. In *Proceedings of the 3rd IEEE Information Assurance Workshop (IASW'02)*, June 2002.

[17] Waikato internet traffic storage. http://wad.cs.waikato.ac.nz/wand/wits/index.html.

[18] N. Weaver, S. Staniford, and V. Paxson. Very fast containment of scanning worms. In *Proceedings of 13 USENIX Security Symposium (Security'04)*, October 2004.

[19] M. M. Williamson. Throttling viruses: Restricting propagation to defeat malicious mobile code. Technical Report HPL-2002-172, HP Laboratories Bristol, June 2002.

[20] J. Wu, S. Vangala, L. Gao, and K. Kwiat. An efficient architecture and algorithm for detecting worms with various scan techniques. In *Proceedings of NDSS'2004*, February 2004.

[21] C. Zou, D. Towsley, W. Gong, and S. Cai. Routing worm: A fast, slective attack worm based on ip address information. Technical Report TR-03-CSE-06, Umass ECE Dept., Novemeber 2004.

[22] C. C. Zou, L. Gao, W. Gong, and D. Towsley. Monitoring and early warning for internet worms. In *Proceedings of ACM CCS'2003*, October 2003.

[23] C. C. Zou, W. Gong, and D. Towsley. Code red worm propagation modeling and analysis. In *Proceedings of ACM CCS'2002*, October 2002.

[24] C. C. Zou, W. Gong, and D. Towsley. Worm propagation modeling and analysis under dynamic quarantine defence. In *Proceedings of ACM WORM'2003*, October 2003.