Effective Worm Detection for Various Scan Techniques

Jianhong Xia, Sarma Vangala, Jiang Wu and Lixin Gao
Department of Electrical and Computer Engineering
University of Massachusetts at Amherst
Amherst, MA 01003
{jxia, svangala, jiawu, lgao}@ecs.umass.edu

Kevin Kwiat
Air Force Research Lab
Information Directorate
525 Brooks Road, Rome, NY 13441
kwiatk@rl.af.mil

***Abstract*— In recent years, the threats and damages caused by active worms have become more and more serious. In order to reduce the loss caused by fast-spreading active worms, an effective detection mechanism to quickly detect worms is desired. In this paper, we first explore various scan strategies used by worms on finding vulnerable hosts. We show that targeted worms spread much faster than random scan worms. We then present a generic worm detection architecture to monitor malicious worm activities. We propose and evaluate our detection mechanism called Victim Number Based Algorithm. We show that our detection algorithm is effective and able to detect worm events before 2% of vulnerable hosts are infected for most scenarios. Furthermore, in order to reduce false alarms, we propose an integrated approach using multiple parameters as indicators to detect worm events. The results suggest that our integrated approach can differentiate worm attacks from DDoS attacks and benign scans.**

I. Introduction

Computer worms are self-propagating malicious codes that spread across networks by exploiting security flaws without human intervention. The threats of these worms have kept increasing, leading to increasing management and maintenance costs. Recent studies ([5], [28], [22] have shown that active worms could infect almost all vulnerable machines in the Internet within several hours or even a few minutes. The Slammer worm [16] is an example of such worms that spread to all potential targets in less than 10 minutes. In order to find vulnerable hosts, a worm can scan the entire IPv4 address space randomly. Such a mechanism of finding vulnerable hosts is called random scan. In this paper, we analyze various scan techniques that help worms spread faster than a random scan worm does. In particular, a worm that scans IP address space selectively, for example, the only IP addresses used in the Internet, can spread faster than a worm that randomly scans the entire IPv4 address space. Slapper ([4], [8]) is an example of such worms that have already caused considerable damage to the Internet. These methods reduce the time wasted on scanning unallocated IP addresses and can dramatically increase the spreading speed of worms. They are easy to implement and pose the most imminent threats to the Internet.

In this paper, we first present and analyze the spread of worms that use various scan techniques. We then propose and evaluate our worm detection algorithm, *Victim Number Based Algorithm*. We show that our algorithm can detect various worm scans before 2% of the vulnerable hosts are infected for most scenarios. Since events such as DDoS attacks or port scans may cause false alarms, we propose an integrated algorithm to reduce false alarms using multiple parameters to differentiate worm scans from DDoS attacks. These parameters include the number of destination addresses scanned and the traffic volume observed in the detection networks.

This paper is organized as follows. Section II introduces related work on worm modeling and detection.

Part of this work has been published in the $11th$ Annual Network and Distributed System Security Symposium (NDSS'04) [26]. This work is supported in part by NSF Grant ANI-0208116, Alfred Sloan Fellowship and Air Force Research Lab. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of National Science Foundation, Alfred Sloan Foundation or Air Force Research Lab.

Section III describes various possible scan techniques and the impact of these techniques on worm spreading speed. In particular, we discuss the class of random scanning worms including selective random scan and routable scan. We also introduce a new worm scan method called *divide-conquer scan*, and analyze its spreading speed. Section IV presents a generic worm detection architecture. Section V describes our Victim Number Based Algorithm for worm detection. Section VI evaluates the performance of our algorithm using traffic traces. To reduce false alarms, we propose an integrated algorithm for worm detection using multiple parameters as indicators in Section VII. We summarize the paper in Section VIII.

II. Background And Related Work

There are a number of studies on analyzing worms and their propagation. Weaver [23] proposed a fast spreading worm, called the Warhol worm, that uses various scan methods to find vulnerable hosts. Staniford et al. [22] systematically introduced the threats of worms and analyzed some well-known worms. In addition, smarter scan methods such as localized subnet, hitlist and permutation scans were discussed. These methods focus on improving the efficiency of the worm scan process.

Several models were used to analyze the spreading of worms. Kephart and White [14] introduced an epidemiological model to measure the dynamics of virus population. Zou et al. [28] proposed a two-factor worm model considering the factors on human countermeasures and congestion caused by worm scan traffic. Chen et al. [5] proposed a discrete time model which adopts parameters such as scan rate, patching rate and death rate. By analyzing the factors that influence the spread of worms, these models give insight into containing worm propagation effectively.

Accurate detection and quick defense are always difficult problems for unprecedented worm attacks. There are a number of detection methods using Internet traffic measurements to detect worms. Based on the locations where decisions are made on worm attacks, these methods can be classified into three categories, i.e., detect worms at end hosts, detect worms on enterprise networks, and detect worms in the global Internet.

As an end host, a single computer can apply various strategies to detect and respond to worm attacks. Somayaji et al. [19] proposed a method on worm detection and defense that uses a system call sequence database to compare with new sequences. If a mismatch is found, then the sequence is delayed. Williamson [25] proposed a method, called Virus Throttle, that checks whether a computer sends a SYN packet to new addresses. If so, the packet will be delayed for a short period of time. These two methods can reduce the impact of false alarms because they adopt the strategy of delaying connections instead of dropping packets. Another example of delaying worm spread is the LaBrea tool [15] designed by Liston. This technique reduces the worm spreading speed by holding TCP sessions with worm victims for a period of time. Spitzner [20] introduced the idea of Honeypots, which are hosts that pretend to own a number of IP addresses and passively monitor packets sent to them. By analyzing the scan packets, Honeypots can detect worm attacks and generate the signatures to fight back worm attacks.

To detect worm attacks on enterprise networks, Cheung [6] proposed an activity-graph based detection algorithm that uses the scan activity-graph inferred from the traffic, in which the senders and receivers of packets are the vertices and the relations among them are the edges. This method assumes that the activity graph can be large for a very short period of time during worm attacks. Recent proposals for scan detection on enterprise networks include the works by Jung et al. ([12], [13]) and Staniford et al. ([24], [21]). Jung et al. [13] presented an algorithm for portscan detection on enterprise networks using reverse sequential hypothesis testing. In this algorithm, a random walk based mechanism observes the number of unsuccessful scans being sent out of the enterprise network by a particular host. Hosts that do not receive acknowledgements for a large number of connection attempts are contained by using a rate based blocking technique. This method is further augmented and implemented by a hardware system in [24]. Staniford et al. [24] showed that their system is able to contain malicious activity before a fixed number of scans are sent out of the enterprise network. Gu et al. [10] discussed worm containment in local network using a small address space. Dagon et al. [7] discuss a multi-parameter based worm detection mechanism using honeypots.

In terms of worm detection in the global Internet, Zou et al. [27] explored the possibility of monitoring Internet traffic with a small address space to predict the worm propagation in the Internet. They use Kalman filter based technique to detect the existence of worm scans. However, Kalman filter based approach is found to be sensitive to the selection of parameters such as monitoring time intervals [10].

Since the number of infected machines increases dramatically and exhibits the trend of exponential growth, we are able to monitor the change on the number of infected machines to detect worms. We call the infected machines observed in our monitoring networks *victims*.

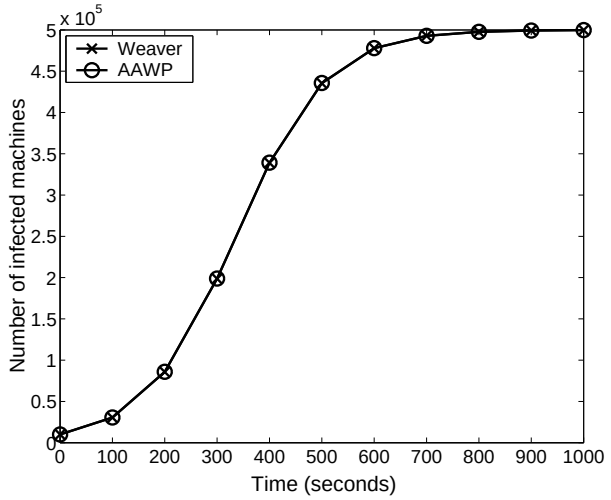


Fig. 1. Comparison between AAWP model and Weaver's simulator for a worm with scan rate=100/sec, number of vulnerable machines=500,000 and hitlist size=10,000. Note that the curves of these two models overlap.

The number of victims is directly related to the number of infected machines in the Internet and the size of monitoring networks, which enables us to capture worm events when the number of victims increases steadily. In this paper, we propose an algorithm, *Victim Number Based Algorithm*, to detect worm events in the Internet.

III. Worm Scan Techniques

In order to propagate itself in the Internet, a worm needs to find vulnerable machines and then infect them. To find vulnerable machines, a worm can either simply scan the entire IPv4 address space randomly, or may perform various strategies to scan the entire or partial IPv4 address space to find targeted hosts. In this section, we discuss various scan strategies and analyze their spreading speed.

A. Random Scan

A worm randomly searches the entire IPv4 address space, which contains 2^{32} possible IP addresses, to find vulnerable machines. We call such scan method *random scan*. There are two existing models to simulate the random scan worm propagation. One is the epidemiological model proposed by Kephart et al. [14], and the other is AAWP model proposed by Chen et al. [5]. Due to the equivalence of these two models as shown in Fig. 1, we adopt the AAWP model in this paper. Based on the AAWP model, the spread of worm is characterized as follows:

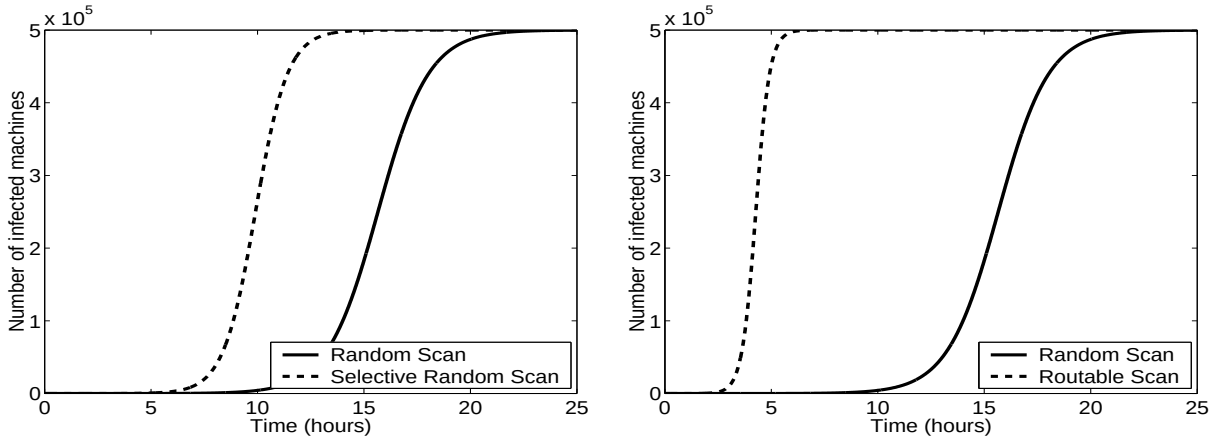
$$n_{i+1} = n_i + [N - n_i][1 - (1 - \frac{1}{\Omega})^{sn_i}] \qquad (1)$$

where N is the total number of vulnerable machines in the Internet; Ω is number of the addresses that a worm performs random scan; s is the scan rate (the number of scan packets sent out by an infected machine per time tick) and n_i is the number of infected machines up to time tick i. These notations are used consistently throughout this paper. In Equation (1), the first term on the right hand side denotes the number of infected machines alive at the end of time tick i. The term, $N-n_i$, denotes the number of vulnerable machines not infected by time tick i. The remaining term, $1-(1-\frac{1}{\Omega})^{sn_i}$, is the probability that an uninfected machine will be infected at the end of time tick $i+1$. We do not consider the death rate due to computer crash and patching rate due to maintenance here. Code Red [17] is a typical example of random scan worms.

B. Selective Random Scan

Instead of scanning the entire IPv4 address space blindly, a worm can scan the partial IPv4 address space that is more likely to be used in the Internet. This will help the worm spread faster by reducing the waste of time on scanning unallocated addresses. The selected address list can be obtained from other resources such as IANA's IPv4 address allocation map [11]. Such scan technique with target selection is called *selective random scan*. The Slapper worm [4] has used this scan technique to spread rapidly. However, worms using the selective random scan need to carry information about the selected target addresses. Carrying such information enlarges the worm's code size and slows down the spreading and infection processes. This information can be hundreds of bytes long and therefore, may not provide much advantage over the random scan.

To compare the spreading speed between random scan worms and selective random scan worms, we do not consider such additional payload information on selected target addresses. Fig. 2(a) compares the spreading speed of worms that use random scan and selective random scan techniques. The parameters are chosen as the same for both the random scan and the selective random scan. The total number of vulnerable machines N is 500,000; the scan rate s is 2 scans/second. The random scan worms use the entire IPv4 address space which has about $2^{32} \approx 4.3 \times 10^9$ addresses. The selective random scan worms use only 162 /8 address blocks which contain about 2.7×10^9 addresses. Fig. 2(a) demonstrates that worm can spread much faster using a selective address pool than using the entire IPv4 address space.



(a) Random Scan vs. Selective Random Scan.

(b) Random Scan vs. Routable Scan.

Fig. 2. Spreading speed of random scan, selective random scan and routable scan.

C. Routable Scan

In order to further reduce scanning address space, a worm may avoid scanning the address space that could not be routed in the Internet. It means that a worm can obtain all routable addresses as scan targets in order to spread fast and effectively. We call this technique *routable scan*. However, this worm has to carry a database of routable IP addresses in its code. The size of this database will affect the propagation speed. A database of larger size will lead to a longer infection time, resulting in slower worm propagation. In this subsection, we will discuss how the size of database can be reduced and analyze the spread speed of routable scan worms.

1) Design of Routable Scan Worm: BGP [18] is an interdomain routing protocol that glues independent networks together. Routable prefixes can be obtained from BGP routing tables. This information enables worm designers to reduce the scanning address space by removing those addresses that cannot be routed. However, carrying a large size of routable prefixes in a portable code of worm might not be efficient and feasible. To ensure that a worm carries a small size of routable prefixes while those prefixes cover as many IP addresses, we present an approach to reducing the size of payload while keeping as much information as possible. The approach consists of the following three steps:

- Step 1. We collect routable prefixes in the BGP tables from RouteViews servers [1]. There are about 112K prefixes. We remove the prefixes that are more specific if their supernets exist. After removing such prefixes, only 49K prefixes are left.
- Step 2. We merge contiguous address segments into a larger address segment. For example, the prefixes 3.0.0.0/8 and 4.0.0.0/8 can be merged into a new segment from 3.0.0.0 to 4.255.255.255. After merging continuous address segments, we reduce the total number of address segments to 17,918, which covers about 1.17×10^9 IP addresses. Comparing this with the total number of IPv4 addresses, we find that around 27.3% of the IPv4 addresses are routable.
- Step 3. We combine address segments using a distance threshold. That is, if two address segments are close to each other in terms that their distance is less than the predefined threshold, 65536, we combine them into a new segment. After combining address segments, the number of address segments is reduced to 1926, which covers about 1.31×10^9 IP addresses.

To store the address segments, a database of about 15.4K bytes (each entry needs 8 bytes) is required. Furthermore, by analyzing the size distribution of the address segments, we find that the largest 20% segments contribute to over 90% of the covered IP addresses. Therefore, a database of the largest segments within 3K bytes can still cover about 90% of the routable IP addresses.

2) Spreading Speed of Routable Scan Worm: From the analysis above, we know that the worm that employs routable scan needs to scan only 10^9 IP addresses instead of 2^{32} addresses, which is four-fold smaller. Hence, routable scan worm has a scanning space of size $\Omega \approx 10^9$. For other parameters, we use the same settings as random scan. Fig. 2(b) shows the spreading speed of routable scan and random scan. We find that if random scan worm needs to spend about 24 hours to infect almost whole vulnerable machines, the routable scan worm only needs to spend about 7 hours to do it. Clearly, routable scan strategy greatly increases the

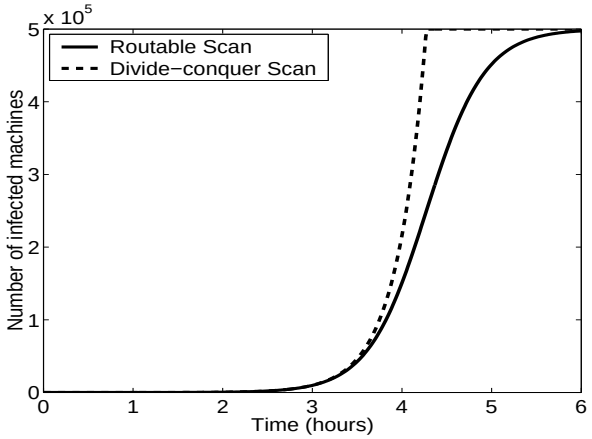


Fig. 3. Spreading speed of divide-conquer scan.

worm spreading speed.

D. Divide-Conquer Scan

Instead of scanning the complete address space in the routable database from each infected host, an infected host can divide its scanning address space among its targets. We call it *divide-conquer scan*. For example, after machine A infects machine B, machine A will divide its routable addresses into halves and transmit one half to machine B. Then machine A scans one half of scanning address space and machine B scans the other half. Using divide-conquer scan, the code size of the worm can be further reduced, because each new infected machine scans a different and smaller address space.

One drawback of divide-conquer scan is single point of failure. During worm propagation, if one infected machine is turned off or gets crashed, then the database passed on to it will be lost. The earlier this happens, the larger the impact will be. There are several measures to address this problem.

One possible method is the generation of a hitlist, where a worm infects a large number of hosts before splitting the database. Another possible method is the use of a generation counter. Each time the worm program is transferred to a new victim, a counter is incremented. The worm program decides to split the database based on the value of the counter. A third possible method is to randomly decide whether or not to divide the database.

To understand the spreading speed of divide-conquer scan worm, we use AAWP model to characterize its spread. As we know, when an infected host finds a vulnerable host, it will divide its routable addresses into halves and transmit one half to the vulnerable host. It means that, at any time, each infected host uses a different scanning space. For simplicity, we assume that the entire space of routable addresses is equally divided on all infected hosts at any time tick. Thus, we have the following equation to model the spreading of divide-conquer scan worms.

$$n_{i+1} = n_i + (N - n_i)[1 - (1 - \frac{1}{\Omega/n_i})^s] \qquad (2)$$

where $i \geq 0$. From Equation (2), we see that the spreading speed of divide-conquer scan may be slightly faster than that of routable scan. When $\frac{sn_i}{\Omega} \ll 1$, the spreading speed of divide-conquer scan is quite close to the spreading speed of routable scan.

However, one of the advantages of using divide-conquer scan is that this worm can be designed to scan any network address no more than once. For example, when an infected host A scans a network address d, no matter the infection is successful or failed, the infected host A will remove the address d from its scanning space to avoid further scanning. Using this strategy, any routable address is scanned at most once. Such a strategy will make the spreading of divide-conquer scan worms more efficient. In this study, we consider divide-conquer scan worms using this strategy to speed up the propagation. To understand the spreading speed of such divide-conquer scan worms, we now have the following spreading model for the divide-conquer scan:

$$n_{i+1} = n_i + (N - n_i)[1 - (1 - \frac{1}{\Omega_i/n_i})^s] \qquad (3)$$

where Ω_i denotes the number of routable addresses that have not been scanned up to time tick i, which can be expressed as follows,

$$\Omega_i = \begin{cases} \Omega - \sum_{j=0}^{i-1} n_j s, & \text{if } \Omega - \sum_{j=0}^{i-1} n_j s \geq n_i \\ n_i, & \text{otherwise} \end{cases}$$

We choose the same setting of parameters with routable scan worms, and compare the spreading speed of divide-conquer scan worms with routable scan worms. Fig. 3 shows that the divide-conquer scan is much faster than the routable scan, although the spreading process is more complicated.

E. Comparison of Various Scan Techniques

The basic difference among various scan methods lies in the selection of the address database. The larger size of selected address database for scanning, the slower worms can spread. For example, selective random scan worm has a smaller scanning address space than random scan worm, which makes it spread faster. Routable scan worm reduces the scanning address space further than selective random scan worm, and it improves spreading speed again. For divide-conquer scan worm, it removes the

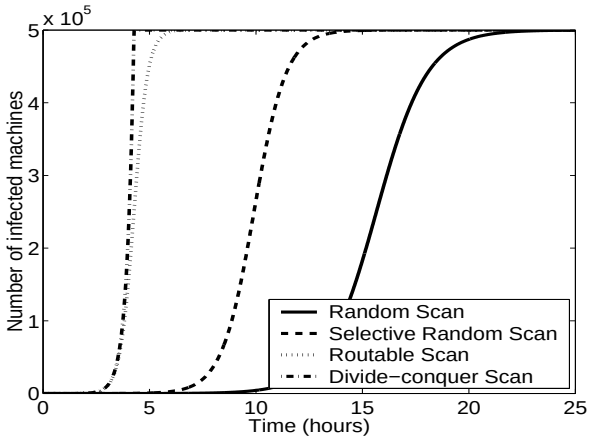


Fig. 4. Comparison of various scan methods.

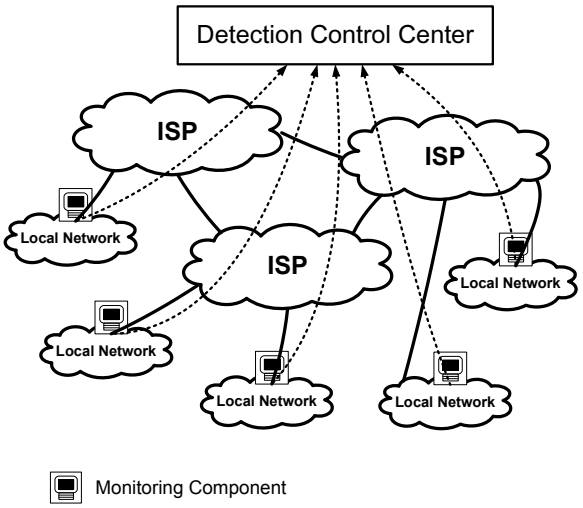


Fig. 5. A Generic Worm Detection Architecture

network addresses that have already been scanned, splits the scanning address space every time when it affects new vulnerable machines. Therefore, it has the fastest spreading speed among these worms. Fig. 4 compares the spreading speed of worms with random scan, selective random scan, routable scan and divide-conquer scan. It clearly shows that the divider-conquer scan worm spreads the fastest among these scan methods.

IV. Architecture for Worm Detection

In order to detect scanning worms, we need to observe various anomalies that are most likely caused by worms. These anomalies can be observed either at end hosts, on local networks, or in the global Internet. The advantage of observing anomalies from the global Internet is that we can detect worm faster and differentiate the worm scans from local events. In this section, we present a generic architecture for worm detection in the global Internet.

A. A Generic Worm Detection Architecture

Monitoring traffic towards a single network is often not enough to detect a worm attack. This is because worms may have already spread widely in the Internet but have not infected the monitored network yet, or worms may never infect the monitored network at all. Therefore, we need to deploy multiple monitoring points on various networks and aggregate the information thus obtained. To achieve this, we propose a distributed worm detection architecture. The architecture monitors the network behavior at different places. By gathering information from different networks, a detection control center can determine the presence of a large scale worm attack. Problems such as where the monitors should be deployed, what needs to be monitored in the network and how the information obtained by monitoring should be aggregated, have to be considered in designing the detection architecture.

We propose a generic traffic monitoring and worm detection architecture as shown in Fig. 5. The architecture is composed of a detection control center and a number of monitoring components. The monitoring components pre-analyze the traffic and send preliminary results or alarms to the detection control center. The detection control center collects these reports from the monitoring components and makes the final decision on whether there is anything serious happening. To avoid single point of failure and to reduce the overload of control center, we may have multiple detection control centers to share the load of computation and communication. In this paper, we focus on evaluating the performance of our system for worm detection, and will not discuss about the detailed design and implementation of the detection control center and monitoring components.

B. Deployment of Monitoring Components

The detection networks consist of a set of addresses monitored by monitoring components. The monitoring components can be deployed on virtual machines or on gateways of local networks. They can also be traffic analyzers beside routers, observing the traffic to a set of addresses.

Worms may not choose to scan the entire IPv4 address space. In addition, different worms might use different target spaces. To make packet collection efficient and effective, the monitoring components should be deployed such that detection networks cover the address space as much as possible, so that the probability of observing the worm event can be high.

To identify worm scan traffic from normal traffic, we consider any packet destined to an inactive address as malicious scan packet which may be caused by worms. Inactive addresses refer to the addresses that have been allocated but have not been assigned to any physical devices or computers. Inactive addresses can be collected and monitored on the individual networks. We should note that, besides the traffic towards to inactive addresses, we can also collect the traffic to active addresses that do not provide requested services. For example, most dial-up users do not provide HTTP server services. Any traffic that tries to access these addresses with HTTP services can be considered as malicious scans. Therefore, our architecture does not need to rely on the large number of inactive addresses only. Considering the traffic to active addresses but requesting inactive services will offer our system better chances to collect malicious traffic for worm detection.

V. Victim Number Based Algorithm for Worm Detection

Since worm signature is not known beforehand, we need to detect anomalies that are most likely caused by worms. Using our detection architecture, we need to design algorithms to detect such anomalies. Serious worm incidents usually involve a large number of hosts scanning specific ports on a set of addresses. Because it is hard for worms to obtain the list of all vulnerable machines in the Internet beforehand, worms normally need to randomly search for targets to infect. Such random scanning techniques will induce a large number of packets to inactive addresses or inactive services. If we detect a large number of distinct addresses sending scan packets to inactive addresses or inactive services within a short period of time, then it is highly possible that there is a worm attack.

We define the source addresses that attempt to connect to inactive address as *victims*. Our detection system will track the victims observed from all monitoring components. The control center will determine whether there is a worm attack based on the change of victim number. Worm detection based on the change of victim number can be considered as a change-point detection problem. Similar to the typical sequential change-point detection algorithms such as parametric or nonparametric Cumulative Sum (CUSUM), our Victim Number Based Algorithm calculates the change on the number of victims and compares it with an adaptive threshold to detect worm events.

A. Victim Decision Rules

To detect the change on the number of victims, we need to identify which source addresses are victims. One of the simplest rules is that, if a source address sends at least one scan packet to an inactive address, we consider this source address a victim. We call this rule *One Scan Decision Rule* (OSDR).

Though very simple, OSDR is susceptible to daily scan noises. For example, when a legitimate user mistypes a destination address, the source address might be marked as a victim if the mistyped destination address is inactive. To avoid such scan noises, we adopt *Two Scan Decision Rule* (TSDR), that is, if a source address sends at least two scan packets to inactive addresses, we will consider this source address a victim. TSDR works well with noise and reflects the incessant feature of worm scans, but it needs to keep track of the number of scans to inactive addresses for each source address, which leads to a more complicated and expensive implementation than OSDR. However, other techniques such as Bloom Filter can be used to alleviate the complexity on the implementation of TSDR.

B. Adaptive Threshold

In our Victim Number Based Algorithm, we use an adaptive threshold to detect anomaly. When the number of new victims is greater than the adaptive threshold T_i in Equation (6), we consider there is an anomaly.

$$\Delta V_{i+1} - E[\Delta V_i] > T_i \tag{4}$$

$$E[\Delta V_i] = \frac{1}{k} \sum_{j=i-k}^{i-1} \Delta V_j \tag{5}$$

$$T_i = \gamma * \sqrt{\frac{1}{k} \sum_{j=i-k}^{i-1} (\Delta V_j - E[\Delta V_i])^2} \tag{6}$$

where V_i is the number of victims detected by the system up to time tick i. $\Delta V_{i+1} = V_{i+1} - V_i$, which denotes the number of new victims detected from time tick i to time tick $i+1$. $E[\Delta V_i]$ is the average number of new victims over last k time ticks at time tick i, and k is the learning time of the system. γ is a constant value called threshold ratio. T_i is the adaptive threshold at time tick i.

To reduce the false positive rate, in practice, we also need to observe a number of such anomalies to determine worm activity. The number of consecutive times of anomaly needed to detect worm activity is denoted as r. A tradeoff exists in the selection of the value of r. A larger value of r gives a lower false positive rate but takes longer time to detect worms whereas a smaller value of

Victim Number Based Algorithm:

1. Gather scan packets using detection architecture.
2. Identify victims using TSDR.
3. Set number of consecutive times that anomalies are observed r, learning time k and threshold ratio γ.
4. Set adaptive threshold T_i for the current time tick i.
5. **do**
   if $\Delta V_{i+1} - E[\Delta V_i] > T_i$ **then**
   $count = count - 1$;
   else
   $count = r$;
   end if
   Update threshold T_i for the current time tick i.
6. **while** $(count > 0)$
7. Alert a worm attack.

Fig. 6. Victim Number Based Algorithm for Worm Detection

r may result in a larger false positive rate but takes less time to detect worms.

In order to smooth the initial learning process, we need to deploy some schemes to expire the entries in the database. A simple method is to use new database everyday. For example, the learning process will start from what the database learned from the previous day. Another method is to assign a decreasing life time L to each new victim detected. If L decreases to zero then the victim is considered as expired and removed from the victim list. If a scan packet is received from the victim before L expires, its lifetime is then reset to L. Using this method, the size of the database can be kept stable. However, keeping track of the timers for each address is expensive. We use the method with daily reset for our solution.

The Victim Number Based Algorithm is as shown in Fig. 6. The monitoring components gather scan packets to the detection networks, and use TSDR to identify the victims. The detection control center collects the victims from all monitoring components and performs Victim Number Based Algorithm to detect whether or not there is a worm.

VI. Performance of Victim Number Based Algorithm

Before we evaluate our detection algorithm, first we need to understand how the number of victims increases during worm events given a detection network size, which will guide us to choose the desired size of detection network. Then we need to set the parameters including the learning time, the threshold ratio constant and the number of consecutive times that anomalies are observed. We choose these parameters based on the properties of the background traffic. In this section, we use traffic traces to decide the parameters and evaluate our detection algorithm.

A. Modeling the Number of Victims

Predicting the number of victims during worm events is important for us to detect the abrupt changes. For worms with random scan, selective random scan and routable scan, we use AAWP model to model the number of victims as follows when OSDR is applied.

$$V_k = \sum_{i=0}^{k} (n_i - n_{i-1})[1 - (1 - \frac{D}{\Omega})^{(k-i)s}] \qquad (7)$$

where V_k is the number of victims detected by the system up to time tick k, n_i is the number of infected hosts up to time tick i, D is the detection network size, i.e., the total number of the inactive addresses being monitored, Ω is the number of addresses that worms perform scan, and s is the scan rate.

Similarly, when we use TSDR to determine victims, the number of victims can be modeled as the following equation,

$$V_k = \sum_{i=0}^{k} (n_i - n_{i-1})[1 - \rho^{(k-i)s} - \rho^{(k-i)s-1}(1-\rho)(k-i)s] \qquad (8)$$

where $\rho = (1 - \frac{D}{\Omega})$. The total number of victims detected at the end of time tick k should be the sum of new victims detected at every time tick i before k. On the right hand side of the equation, $(n_i - n_{i-1})$, is the number of newly infected machines during time tick i. $\rho^{(k-i)s}$ denotes the fraction of infected machines during time tick i that have never scanned the addresses in the detection network up to time tick k. The term $\rho^{(k-i)s-1}(1-\rho)(k-i)s$ denotes the fraction of infected machines during time tick i that have scanned the addresses in the detection network only once up to time tick k. Equations (7) and (8) assume that each newly infected host uses the same address space to do further scan.

However, for worms with divide-conquer scan method, each infected host has a different scanning address space. The worms will not scan any network address more than once. Hence, after a certain period of time, all addresses in the detection network will be scanned by worms, and no more victims can be monitored by the detection network. To simplify the analysis, we assume that, at any time before all detection network addresses have been scanned, the number of detection network addresses that have not been scanned is proportional to the number of routable addresses that have not been scanned. Similar to

Equation (8), using TSDR to detect divide-conquer scan worms, the number of victims detected by the system up to time tick k can be expressed by the following equation:

$$V_k = \begin{cases} \sum_{i=0}^{k}(n_i - n_{i-1})[1 - \alpha_{k,i} - \beta_{k,i}], & \text{if } \frac{\Omega_k D}{\Omega} \geq 1 \\ V_{k-1}, & \text{otherwise} \end{cases}$$

where $\alpha_{k,i} = (1 - \frac{D}{\Omega})^{(k-i)s}$, which denotes the fraction of infected machines during time tick i that have never scanned the addresses in the detection network up to time tick k. $\beta_{k,i} = (1-\frac{D}{\Omega})^{(k-i)s-1}(\frac{D}{\Omega})(k-i)s$, which denotes the fraction of infected machines during time tick i that have scanned the addresses in the detection network only once up to time tick k. Ω denotes the total number of routable addresses. $\Omega_k = \Omega - \sum_{j=0}^{k-1} n_j s$, which denotes the number of routable addresses that have not been scanned up to time tick k. D is the detection network size.

B. Requirements for Detection Network Size

Our detection system monitors the traffic to a set of inactive addresses and determines the number of victims in the Internet. The more inactive addresses we monitor, the closer the number of victims is to the real number of infected hosts in the Internet. Thus, the detection network size, i.e., the total number of monitored inactive addresses, is important for tuning the performance of our algorithm.

To understand the impact of the detection network size, we analyze the difference on the number of victims we observed from the detection network and the number of infected hosts in the Internet. Without considering the noises on the background traffic, we use AAWP model to analyze the number of infected hosts in the Internet and the number of victims monitored in detection networks using TSDR rule for various worm scan methods.

Fig. 7 shows the results on different detection network sizes for various scan methods. Fig. 7(a) shows the difference on the number of observed victims and the number of infected hosts in the Internet for random scan worms. We see that the number of victims detected approaches the number of infected hosts when the detection network size is over 2^{20} (a /12 network). For routable scan in Fig. 7(b), we have the similar result as random scan except that the spreading speed is faster than that of random scan.

For divide-conquer scan in Fig. 7(c), we can see that when a /8 detection network is used for divide-conquer scan, the number of victims detected approaches the number of infected hosts closely before all addresses in detection network have been scanned. When a /12 detection network is used, the number of victims detected lags a little behind the infection curve. However, when a /16 detection network is used, all addresses in the detection network will be scanned soon, and the number of victims lags far behind the infection curve. This suggests that, comparing with random scan and routable scan, it is difficult to detect infected hosts of divide-conquer scan worms as victims using TSDR when the detection network size is not large enough.

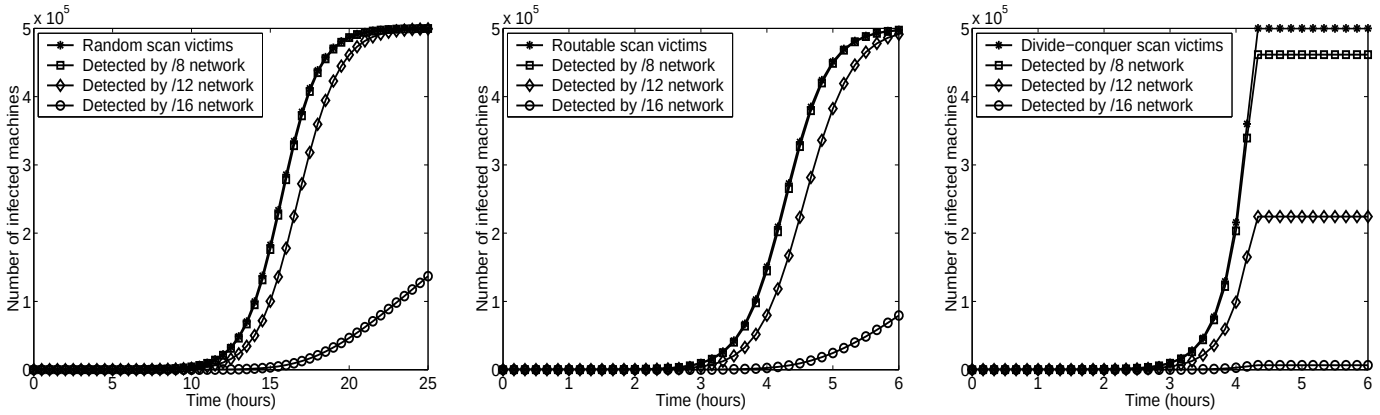
Although the infection dynamics of random and routable scan methods can be captured with comparable performance, divide-conquer scan is very hard to detect. Fortunately, divide-conquer scan has its limitations and attackers often combine it with random scan to enhance worm spreading speed, which makes our solution possible to detect worms in that particular case.

C. Traffic Collection

We evaluate our algorithm on real traffic traces from worm incidents. However, it is very difficult to find such Internet traffic traces that are publicly available. In this study, we use traffic traces from the WAND research group [9]. These traffic traces are gathered from the gateways on the campus network at the University of Auckland, New Zealand, who owns a /16 prefix. We use an incoming traffic trace recorded on June 12, 2001, because the date is close to the day when Code-Red I V2 broke out (July 19, 2001). The packets we are interested in are the SYN packets sent towards TCP port 80 (HTTP). Using worm infection dynamics from Equation (8), we add the simulated number of worm victims detected by a /14 network over time into this real traffic traces. Combining the victim number dynamics from the real incoming traffic with the simulated worm traffic, we get the simulated victim number dynamics on the network under different worm attacks. In the simulation, we assume that the worm starts at 3:00am in the morning. We use these traces to evaluate our detection algorithm.

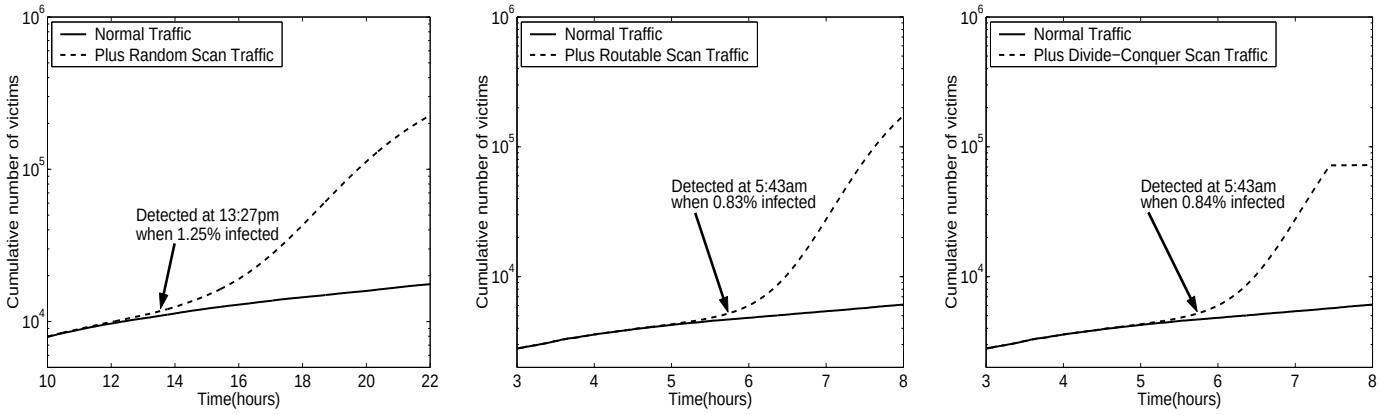
D. Parameter Selection

Selecting appropriate parameters for the detection system is important. We try to select parameters to achieve small false alarm rates. We perform an exhaustive search in the domains of the learning time k, the threshold ratio γ and the number of consecutive times that anomalies are observed r. Intuitively, the learning time, k, cannot be too long as it can lead to a longer detection time. Small value of k may lead to false alarms due to background noises. We also find that a larger value of r can reduce false alarms, but it will take longer time to detect worm



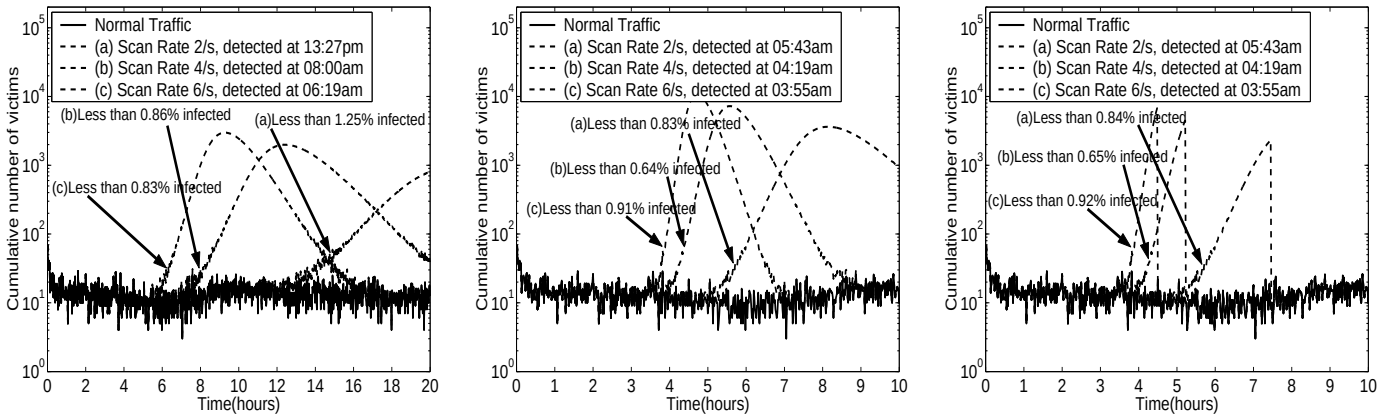
(a) Random Scan (b) Routable Scan (c) Divide-conquer Scan

Fig. 7. Impact of detection network size on observed number of victims.



(a) Detection of a random scan (b) Detection of a routable scan (c) Detection of a divide-conquer scan

Fig. 8. Detection time for random scan, routable scan and divide-conquer scan worms using a /14 detection network.



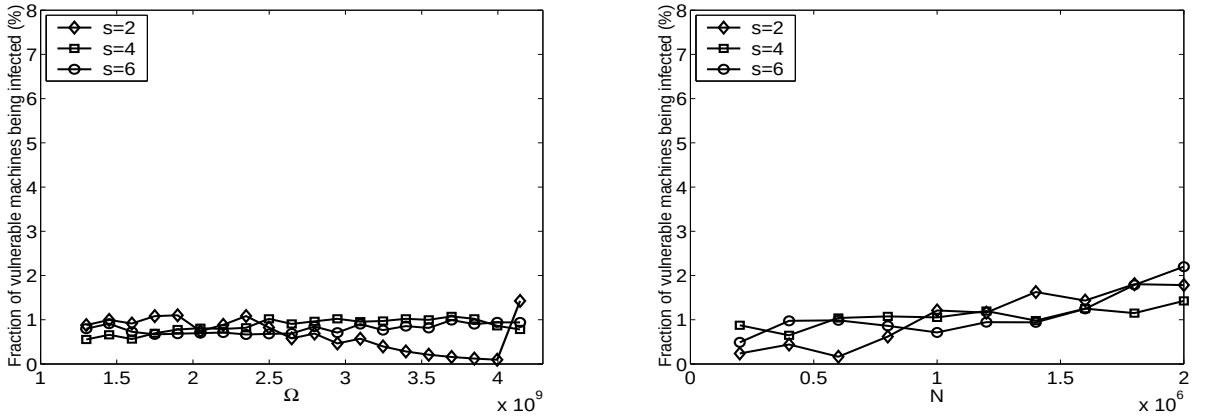
(a) Detection on random scan worms (b) Detection on routable scan worms (c) Detection on divide-conquer scan worms

Fig. 9. Detection on random scan, routable scan and divide-conquer scan worms with various scan rates.

attacks. A larger value of γ makes worm detection more difficult, but it will have less false alarms. From our experiments, we choose $k = 240$, $\gamma = 3$ and $r = 10$ in our evaluation study.

E. Evaluation of Victim Number Based Algorithm

To evaluate our algorithm on real traces, we combine the real trace traffic with simulated worm traffic based on various random scan methods. Fig. 8(a) shows the detection time for random scan worm. The worm starts



(a) Fraction of vulnerable machines being infected vs. Ω.

(b) Fraction of vulnerable machines being infected vs. N.

Fig. 10. Fraction of vulnerable machines being infected when worm events are detected, varying Ω and N.

at 3:00am in the morning with scan rate of 2 per second and is detected at 13:27pm when less than 1.25% of vulnerable machines are infected. It shows that with the /14 network, there is a rapid increase in the number of victims during random scan worm attacks. Fig. 8(b) shows the case when worms perform routable scan. We can see that when worms perform routable scan, we detect worm events at 5:43am. At this time, less than 0.83% of vulnerable machines are infected. For divide-conquer scan, as shown in Fig.8(c), we have similar results as routable scan because the changes on the number of victims for both scan methods are similar during the early stage of worm spreading. However, the spreading speed of divide-conquer scan is faster than routable scan. When we detect divide-conquer scan worm at 5:43am, less than 0.84% of vulnerable machines are infected.

Besides the various types of scan methods, we want to know to what extent the victim number based detection algorithm works for worms with different scan rates. Fig. 9(a) gives the results on the fraction of vulnerable machines that have been infected when our algorithm detects worm events by varying scan rates using a /14 detection network. The Y-axis shows the number of new victims detected in each time interval. We can see that our algorithm can detect worms with higher scan rates earlier than worms with lower scan rates. Fig. 9(b) and Fig. 9(c) show similar plots for routable scan and divide-conquer scan worms respectively.

To understand how Ω (the number of addresses that a worm performs random scan) and N (the number of vulnerable machines in the Internet) affect the performance of our algorithm, we look at various cases varying these numbers and check the fraction of vulnerable machines that have been infected when we detect worm events. In Fig. 10(a), we vary Ω from 1.3×10^9 to 2^{32} when $N = 500,000$. The worm can be detected before 1.4% of vulnerable machines are infected in most cases. In Fig. 10(b), we vary N from 0.1×10^6 to 2.0×10^6 when $\Omega = 2^{32}$. It shows that worms can be detected before 2% of vulnerable machines are infected.

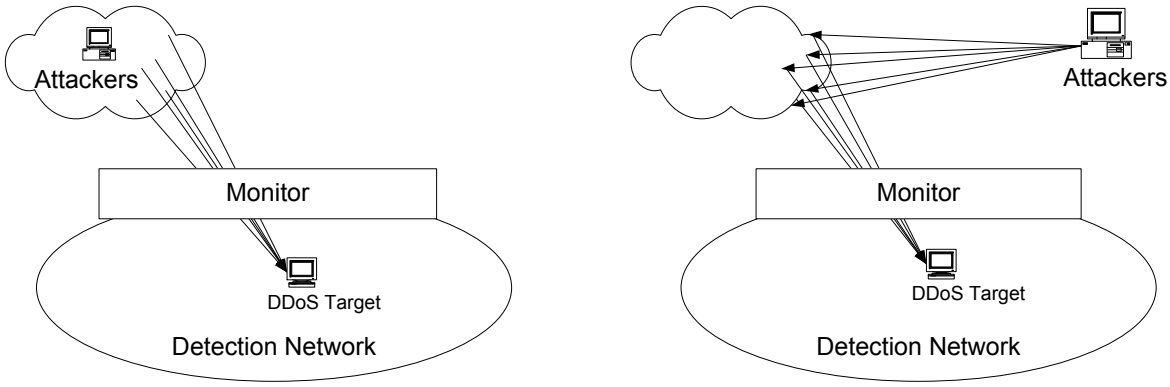
VII. Integrated Algorithm For Reducing False Alarms

As shown from the previous section, the victim number based algorithm is able to detect worm events involving random and routable scans when the number of victims increases. However, our detection system may also observe that the number of victims increases during DDoS attacks, which can lead to false alarms. In this section, we analyze the performance of victim number based algorithm for scenarios where it is prone to yield false alarms. We alleviate the problem of false alarms by analyzing other indicators during a worm attack, and propose an integrated algorithm for worm detection.

A. Scenarios Causing False Alarms

Here we look at the following two scenarios where our system might yield false alarms:

- **DDoS Attacks**: In the case of a DDoS attack, we could observe a large number of victims. For example, in Fig. 11(a), an attacker could randomly spoof a large number of source IP addresses to attack a target in the detection network. In this case, our monitor will detect the number of victims increasing and trigger an alarm of worm attack. However, it is not a worm attack. As another example in Fig. 11(b), an attacker can also use reflection DDoS attack to make our system trigger false alarms. In this case, the attacker puts a spoofed source address in the packets and sends them to a



(a) Spoofed source DDoS attacks. (b) Reflection DDoS attacks.

Fig. 11. Scenarios causing false alarms, where the number of victims is large but the number of destinations is small.

large number of hosts. The spoofed source address is one of the addresses in our detection network. Therefore, our monitor will observe a large number of replies to the detection network from those hosts. It will result in significant increases in the number of victims and lead to a false alarm. However, in these cases, we find that the destination addresses in the detection network visited by those DDoS traffic are quite limited. During worm attacks, the number of destination addresses visited by worm scans is much larger than that during DDoS attacks. So we can use this indicator to differentiate worm attacks from DDoS attacks.

- **Benign Scans:** Some traffic, such as the traffic from web-crawlers, might be benign scans that are shown in [12]. However, if there are a large number of such scanners operating at the same time, our detection system might see a large number of victims if these scanners attempt to connect to the addresses in the detection network. In that case, our system might have false alarms. However, the traffic volume caused by web-crawlers or benign scanners should be much less than the traffic volume caused by worms. If we consider the volume of scan traffic as an indicator, we can differentiate worm scans from benign scans.

B. Analysis of DDoS Scans and Benign Scans

In order to reduce the false alarms caused by the above scenarios, we need to understand the differences between worm attacks and DDoS attacks or benign scans. Below is the analysis on those differences that differentiate the worm attacks from DDoS attacks or benign scans.

1) Destination Addresses Visited in Detection Networks: During the period of worm attack and DDoS attack, the number of new destination addresses in the detection network visited by worm scanners may be different from that by DDoS scanners. This number and its variation yield valuable information to differentiate a worm attack from a DDoS attack. The reason is that the number of new destination addresses in the detection network visited by infected machines increases rapidly to a large extent during a worm attack, while this number increases more slowly or even remains unchanged during a DDoS attack.

The number of destination addresses visited in the detection network by infected machines during a worm attack can be derived using the following equation:

$$A_{i+1} = A_i + (D - A_i) * (1 - (1 - \frac{1}{\Omega})^{sn_i}) \quad (9)$$

where A_{i+1} is the number of destination addresses visited in the detection network by infected hosts up to time tick $i+1$, D is the detection network size, Ω is the size of the address space being scanned by the worm, s is the scan rate and n_i is the number of infected nodes at time tick i.

2) Scan Traffic Volume: The differentiation of worm attacks and benign scans can be done by looking at the scan traffic volume coming towards the detection network. Scan traffic volume is defined as the number of scan packets that destine to detection network per unit time. In the case of a benign scan, the volume of scan traffic increases more slowly when compared to the volume of scan traffic received by either worm attacks or DDoS attacks. The scan traffic volume generated by worm infected hosts can be derived from the following equation, which will increase linearly with the number of infected hosts.

$$B_{i+1} = \frac{D}{\Omega} sn_i \quad (10)$$

where B_{i+1} is the volume of scan traffic that destine to detection network up to time tick $i+1$.

C. An Integrated Approach to Worm Detection

Based on the above analysis, we introduce two more parameters to indicate worm attacks. One is the number of new destination addresses visited in the detection network by victims. The other one is the volume of scan traffic that destine to detection network. The number of new destination addresses visited in the detection network can be collected as explained below. Each time when a scan packet from victims is received by the detection system, the monitor component checks whether or not the destination address has already been visited. If it is not, then it is added to a list of destination addresses that have been visited. The detection control center gets this information and uses the same aggregation technique as that with the victim number, that is, a threshold based detection to detect anomaly. The number of the new destination addresses visited in the detection network should be significantly large in the case of a worm attack. During DDoS attacks or normal user behavior, this number is expected to be small, which leads to a distinction between a worm attack and a DDoS attack. We can use a similar approach to detect the anomaly on volume of scan traffic.

The complete integrated approach of worm detection is described in Fig. 12. That is, first, we check whether the victim number is found to be higher than its adaptive threshold T_i. Second, we check whether the number of new destination addresses visited in detection networks by victims is higher than its adaptive threshold T_i^a. Third, we check whether the change of traffic volume towards the detection network is higher than its adaptive threshold T_i^b. If all these anomalies happen multiple times that exceed their corresponding limits on the consecutive times, the control center will report a worm attack.

Integrated Algorithm:

1. Gather scan packets using detection architecture
2. Identify victims using TSDR
3. Initialize variables:
   $count = r$, $count_a = r_a$, $count_b = r_b$
4. **do**
   if $\Delta V_{i+1} - E[\Delta V_i] > T_i$ **then**
   $count = count - 1$
   else
   $count = r$;
   end if
   if $\Delta A_{i+1} - E[\Delta A_i] > T_i^a$ **then**
   $count_a = count_a - 1$
   else
   $count_a = r_a$;
   end if
   if $\Delta B_{i+1} - E[\Delta B_i] > T_i^b$ **then**
   $count_b = count_b - 1$
   else
   $count_b = r_b$;
   end if
   Update threshold T_i, T_i^a and T_i^b for time tick i.
   while ($count \neq 0 \parallel count_a \neq 0 \parallel count_b \neq 0$)
5. Alert a worm attack

Fig. 12. Integrated Algorithm for Worm Detection

D. Evaluation of the Integrated Algorithm

In the integrated algorithm, we use a threshold based detection technique to observe the changes on traffic volume and number of the destination addresses visited in the detection network. The parameters k and γ are chosen to be 240 and 3 respectively, which are the same as before. However, r_a and r_b are chosen to be 5 for the change detections on traffic volume and the new destination addresses visited in the detection network.

To evaluate the integrated algorithm, we use the traffic that contains both worm events and DDoS attacks, and compare the results with that of the original Victim Number Based Algorithm. The DDoS traffic trace is obtained from the backscatter data collected by CAIDA [2]. We add the DDoS traffic and the simulated worm traffic into the real traffic. We assume that both worm event and DDoS attack take place at the same time, 3:00am.

Fig. 13 shows the traffic volume of normal traffic, DDoS traffic and worm traffic with random scan method. We see that at 3:00am, the traffic volume increases a lot due to DDoS attack, but the traffic caused by worms does not increase much. After 18:00pm, the traffic volume caused by worms becomes larger than normal traffic.

To detect worm events from the above traffic, we need to calculate the number of victims. Since this data trace is collected by the network telescope [3] of CAIDA (a group of inactive addresses), every source address in the packet received by this network is considered as a victim. Using the original Victim Number Based Algorithm, we can detect worm at 4:00am, one hour after the onset of DDoS attack. Actually the number of victims caused by worms is very small at 4:00am, but the number of victims caused by DDoS attacks is large at that time. The original Victim Number Based Algorithm will give an alert because it detects the abrupt change on the number of victims, which is not caused by worms. Hence the original Victim Number Based Algorithm has a false alarm. However, using the integrated algorithm, we will not detect worm at 4:00am. Fig. 14 shows that the number of new destination addresses visited in the detection network during a DDoS attack is small (at most

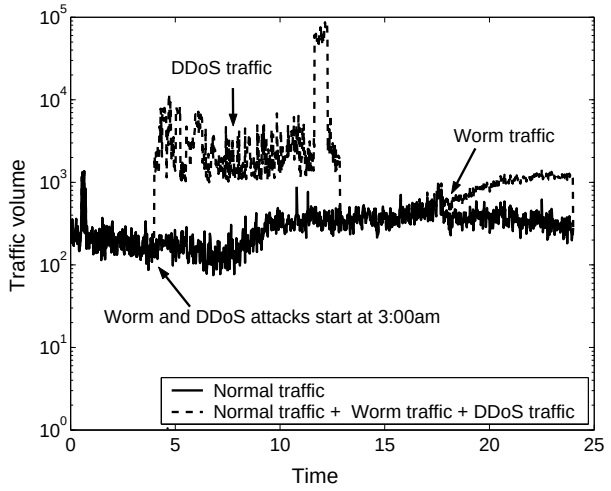


Fig. 13. Volume of normal traffic, DDoS traffic and worm traffic

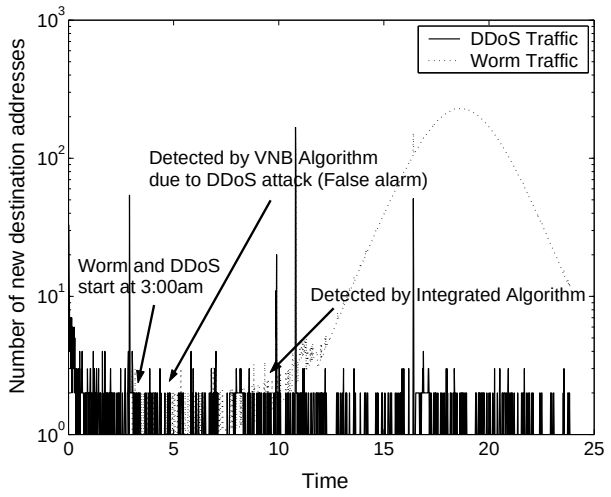


Fig. 14. Number of new destination addresses visited in detection networks during worm attacks and DDoS attacks

1 or 2 new addresses per second). For worm attacks, this number is much large and increases exponentially. The integrated algorithm detects worm scanning at 9:48am, almost 7 hours after the infection starts.

Thus, the integrated algorithm for worm detection improves the performance of the original Victim Number Based Algorithm by reducing false alarms.

VIII. Summary And Future Work

In this paper, we discuss different types of worm scan methods and their effects on future worm propagation, and introduce two new scan techniques, routable scan and divide-conquer scan. These new techniques of worm scans pose a big menace to the network security.

To detect and defend the worm attacks, we propose an algorithm for worm detection on various scan methods, Victim Number Based Algorithm, to detect worms. This algorithm is one of solutions towards the global worm detection system. Its simplicity and effectiveness as shown in this work make it highly practical to implement. Using multiple parameters as indicators of worm attacks, the integrated approach can reduce false alarms by differentiating worm attacks from some DDoS attacks and benign scans.

Although we have investigated the performance and feasibility of Victim Number Based Algorithm in this paper, there are still some issues for future study. For example, we have not analyzed the cost of the implementation of our algorithm in hardware design, and the difficulties on the deployment of monitoring components distributed in the Internet. Our future work includes the estimation of the resources necessary to implement the integrated solution and the study on a viable implementation.

References

[1] University of Oregon Route Views Project. http://www.routeviews.org.

[2] CAIDA. Backscatter telescope data. http://www.caida.org/analysis/security/telescope/.

[3] CAIDA. Telescope analysis. http://www.caida.org/analysis/security/telescope/.

[4] Internet Storm Center. Openssl vulnerabilities. http://isc.incidents.org/analysis.html?id=167, Sept. 2002.

[5] Z. Chen, L. Gao, and K. Kwiat. Modeling the Spread of Active Worms. In *Proc. IEEE INFOCOM*, March 2003.

[6] S. Cheung. Graph-based Intrusion Detection System (GrIDS). http://seclab.cs.ucdavis.edu/projects/arpa/grids/welcome.html.

[7] D. Dagon, X. Qin, G. Gu, W. Lee, J. Grizzard, J. Levine, and H. Owen. HoneyStat: Local Worm Detection Using Honeypots. In *Proceedings of Seventh International Symposium on Recent Advances in Intrusion Detection (RAID 2004)*, September 2004.

[8] F-Secure. Global Slapper Worm Information Center. http://www.f-secure.com/v-descs/slapper.shtml.

[9] The WAND Research Group. Waikato Internet Traffic Storage. http://wand.cs.waikato.ac.nz/wand/wits/index.html.

[10] G. Gu, D. Dagon, X. Qin, M. I. Sharif, W. Lee, and G. F. Riley. Worm Detection, Early Warning, and Response Based on Local Victim Information. In *Proceedings of The 20th Annual Computer Security Applications Conference (ACSAC 2004)*, December 2004.

[11] IANA. Internet Protocol V4 Address Space. http://www.iana.org/assignments/ipv4-address-space/.

[12] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan. Fast Portscan Detection Using Sequential Hypothesis Testing. In *Proceedings of IEEE Symposium on Security and Privacy*, May 2004.

[13] J. Jung, S. E. Schechter, and A. W. Berger. Fast Detection of Scanning Worm Infections. In *Proceedings of Seventh International Symposium on Recent Advances in Intrusion Detection (RAID 2004)*, September 2004.

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

[15] T. Liston. Welcome To My Tarpit - The Tactical and Strategic Use of LaBrea. http://www.hackbusters.net/.

[16] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. The spread of the sapphire/slammer worm. http://www.caida.org/outreach/papers/2003/sapphire/sapphire.html.

[17] David Moore. The spread of code red worm (crv2). http://www.caida.org/analysis/security/ code-red/coderedv2/analysis.xml.

[18] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). Request for Comments 1771, March 1995.

[19] A. Somayaji and S. Forrest. Automated Response Using System-Call Delays. In *Proceedings of 9th Usenix Security Symposium, Denver, Colorado*, August 2000.

[20] L. Spitzner. Strategies and Issues: Honeypots - Sticking It to Hackers. http://www.networkmagazine.com/article/NMG20030403S0005.

[21] S. Staniford. Containment of Scanning Worms in Enterprise Networks. http://www.silicondefense.com/research/researchpapers/scanContainment.

[22] S. Staniford, V. Paxson, and N. Weaver. How to 0wn the Internet in Your Spare Time. http://www.icir.org/vern/papers/cdc-usenix-sec02/.

[23] N. Weaver. Potential Strategies for High Speed Active Worms: A Worst Case Analysis. http://www.cs.berkeley.edu/∼nweaver/worms.pdf.

[24] N. Weaver, S. Staniford, and V. Paxson. Very Fast Containment of Scanning Worms. In *13th USENIX Security Symposium*, August 2004.

[25] M. M. Williamson. Throttling Viruses: Restricting Propagation to Defeat Malicious Mobile Code. http://www.hpl.hp.com/techreports/2002/HPL-2002-172.pdf.

[26] J. Wu, S. Vangala, L. Gao, and K. Kwiat. An Effective Architecture and Algorithm for Detecting Worms with Various Scan Techniques. In *Proceeding of the 11th Annual Network and Distributed System Security Symposium (NDSS'04)*, February 2004.

[27] C. Zou, L. Gao, W. Gong, and D. Towsley. Monitoring and Early Warning for Internet Worms. In *Proceedings in 10th ACM Conference on Computer and Communication Security (CCS'03)*, October 2003.

[28] C. Zou, W. Gong, and D. Towsley. Code Red Worm Propagation Modeling and Analysis. In *Proceedings in 9th ACM Conference on Computer and Communication Security (CCS'02)*, November 2002.