An Effective Architecture and Algorithm for Detecting Worms with Various Scan Techniques

Jiang Wu, Sarma Vangala, Lixin Gao
Department of Electrical and Computer Engineering,
University of Massachusetts,
Amherst, MA 01002.
Email: (jiawu,svangala,lgao)@ecs.umass.edu

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

Abstract

Since the days of the Morris worm, the spread of malicious code has been the most imminent menace to the Internet. Worms use various scanning methods to spread rapidly. Worms that select scan destinations carefully can cause more damage than worms employing random scan. This paper analyzes various scan techniques. We then propose a generic worm detection architecture that monitors malicious activities. We propose and evaluate an algorithm to detect the spread of worms using real time traces and simulations. We find that our solution can detect worm activities when only 4% of the vulnerable machines are infected. Our results bring insight on the future battle against worm attacks.

1 Introduction

A worm is a program that propagates across a network by exploiting security flaws of machines in the network. The key difference between a worm and a virus is that a worm is autonomous. That is, the spread of active worms does not need any human interaction. As a result, active worms can spread in as fast as a few minutes [7][12]. Recent worms, like Code Red and Slammer cost public and private sectors millions of dollars. Furthermore, the propagation of active worms enables one to control millions of hosts by launching distributed denial of service (DDOS) attacks, accessing confidential information, and destroying/corrupting valuable data. Accurate and prompt detection of active worms is critical for mitigating the impact of worm activities.

In this paper, first, we present an analysis on potential scan techniques that worms can employ to scan vulnerable machines. In particular, we find that worms can choose targets more carefully than the random scan. A worm that scans only IP addresses announced in the global routing table can spread faster than a worm that employs random scan. In fact, scan methods of this type have already been used by the Slapper worm[9]. These methods reduce the time wasted on unassigned IP addresses. They are easy to implement and pose the most imminent menace to the Internet. We analyze a family of scan methods and compare them to the random scan. We find that these scan methods can dramatically increase the spreading speed of worms.

Second, we propose a worm detection architecture and algorithms for prompt detection of worm activities. Our worm detection architecture takes advantage of the fact that a worm typically scans some unassigned IP addresses or an inactive port of assigned IP addresses. By monitoring unassigned IP addresses or inactive ports, one can collect statistics on scan traffic. These statistics include the number of source/destination addresses and volume of the scan traffic. We propose a detection algorithm called *victim number based algorithm*, which relies solely on the increase of source addresses of scan traffic and evaluates its effectiveness. We find that it can detect the Code Red like worm when only

[0] This work is supported in part by NSF Grant ANI-0208116, the Alfred Sloan Fellowship and the 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 the National Science Foundation, the Alfred Sloan Foundation or the Air Force Research Lab.

4% of the vulnerable machines are infected.

This paper is organized as follows. Section 2 introduces and analyzes various scan methods. By comparing the spreading speed of various scan methods, we show that routable worms spread the fastest and pose the greatest menace to the Internet. In Section 3, a generic worm detection architecture is introduced. Then, the victim number based algorithm is designed and evaluated. Section 4 concludes the paper.

1.1 Related Work

A lot of work has been done in analyzing worms, modeling worm propagation and designing possible worms. In [18], Weaver designed fast spreading worms, called the Warhol worms, by using various scan methods. In [12], Staniford et. al. systematically introduced the threats of worms and analyzed well-known worms. In addition, smarter scan methods such as local subnet, hitlist and permutation scans were designed. Instead of using the idea of selecting as many assigned addresses as possible, these methods focus on increasing the efficiency of the worm scan process itself.

Many models were designed to analyze the spreading of worms. Kephart and White designed an epidemiological model to measure the virus population dynamics[3]. Zou et. al.[23] designed a two-factor worm model, which takes into consideration the factors that slow the worm propagation. In [20], Chen et. al. designed a concise discrete time model which can adapt parameters such as the worm scan rate, the vulnerability patching rate and the victim death rate. By analyzing the factors that influence the spread of worms, these models and methods give an insight into containing worm propagation effectively.

More important problems are detection and defense. As in the fight against computer viruses, accurate detection and quick defense are always difficult problems for unprecedented attacks. There are a number of detection methods using Internet traffic measurements to detect worms on networks. Zou et.al.[23], explored the possibility of monitoring Internet traffic with small sized address spaces and predicting the worm propagation. Cheung's activity graph based detection[13] uses the scan activity graph (the sender and receiver of a packet are the vertices and the relation between them are the edges) inferred from the traffic. This method uses the fact that the activity graph can be large only for a very short period of time. Spitzner[11] introduced the idea of Honeypots. Honeypots are hosts that pretend to own a number of IP addresses and passively monitor packets sent to them. Honeypots can also be used to fight back a worm's attack. An example is the LaBrea tool[14] designed by Liston, which reduces the worm spreading speed by holding TCP sessions with worm victims for a long time.

Methods to detect worms that cause anomalies on a single computer were also designed. Somayaji and Forrest's defense solution[10] uses a system call sequence database and compares new sequences with them. If a mismatch is found, then the sequence is delayed. Williamson's "Virus Throttle"[19] checks if a computer sends SYN packets to new addresses. If so, the packet will be delayed for a short period of time. By delaying instead of dropping, these two solutions reduce the impact of false alarms. This paper proposes detection methods that can detect large scale worm attacks on the Internet or enterprise networks.

2 Scan Methods

Probing is the first thing that worms perform to find vulnerable hosts. Depending on how worms choose their scan destinations from a given address space, scan methods can be classified as random scan, routable scan, hitlist scan and divide-conquer scan. In this section, we present a set of scan methods and compare their spreading speeds.

2.1 Selective Random Scan

Instead of scanning the whole IP address space at random, a set of addresses that more likely belong to existing machines can be selected as the target address space. The selected address list can be either obtained from the global or the local routing tables. We call this kind of scanning selective random scan. Care needs to be taken so that unallocated or reserved address blocks in the IP address space are not selected for scanning. Worms can avoid using addresses within these address blocks. Information about such address blocks can be found in the following ways. An example is the Bogon list[8], which contains around 32 address blocks. The addresses in these blocks should never appear in the public network. IANA's IPv4 address allocation map[4] is a similar list that shows the /8 address blocks which have been allocated. The "Slapper"[9](or Apache, OpenSSL) worm made use of these lists in order to spread rapidly. Worms using the selective random scan need to carry information about the selected target addresses. Carrying more information enlarges the worm's code size and slows down its infection process. For selective random scan, the database carrying the information can be hundreds of bytes. Therefore, the additional database will not greatly affect the already slow spreading of worms.

2.1.1 Propagation Speed of Selective Random Scan

Two current models used are the Weaver's model in [17] and the AAWP model in [20]. Due to their similar modeling performance as shown in Figure 1, we adopt the AAWP worm propagation model[20]. In the AAWP model the spread of worm is characterized as follows:

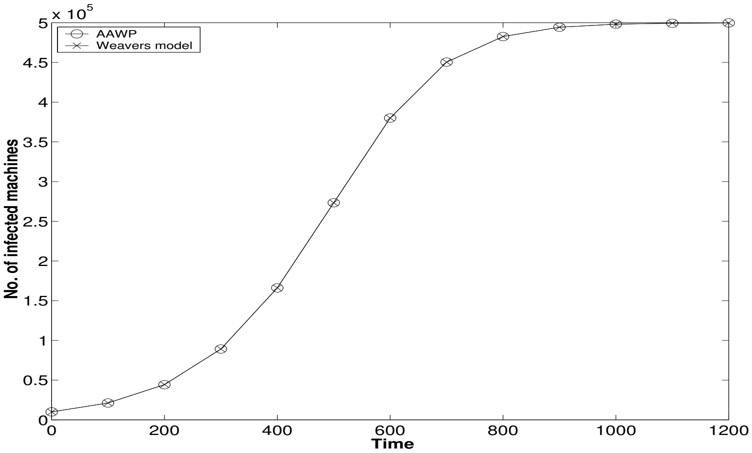


Figure 1. Comparison of AAWP Model and Weaver's Simulator for scan rate=100/sec, no. of vulnerable machines=500,000 and hitlist size=10,000. Note that the curves of the two models overlap.

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

where N is the total number of vulnerable machines; T is the number of addresses used by the worm as scan targets; s is the scan rate (the number of scan packets sent out by a worm infected host per time tick) and n_i is the number of machines infected up to time tick i. The length of the time tick is determined by the time needed to completely infect a machine (denoted by t). 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, $1 - (1 - \frac{1}{T})^{sn_i}$, is the probability that an uninfected machine will be infected at the end of time tick $i + 1$. This model is different from [20] in that the death rate (rate by which infected machines reset or crash) and the patch rate (rate by which vulnerable machines are fixed and become invulnerable) are assumed to be zero. This is because our goal is to find the effect of address space selection on the worm spread.

Figure 2(a), obtained from Equation (1), shows the spreading speed of worms that use random scan and selective random scan techniques. The selective random scan was used by the Slapper worm. Once it infects a machine, the Slapper worm scans only a predefined set of Class A machines[22]. The total number of vulnerable machines (N), the scan rate (s) and t are set to 500,000, 2 scans/second and 1 second respectively for both the random scan and the selective random scan. The Slapper worm which uses selective random scan has 2.7×10^9 addresses as scan targets (T). Figure 2(a) demonstrates that using a target address pool that is smaller than the whole IP address space can speed up the worm propagation.

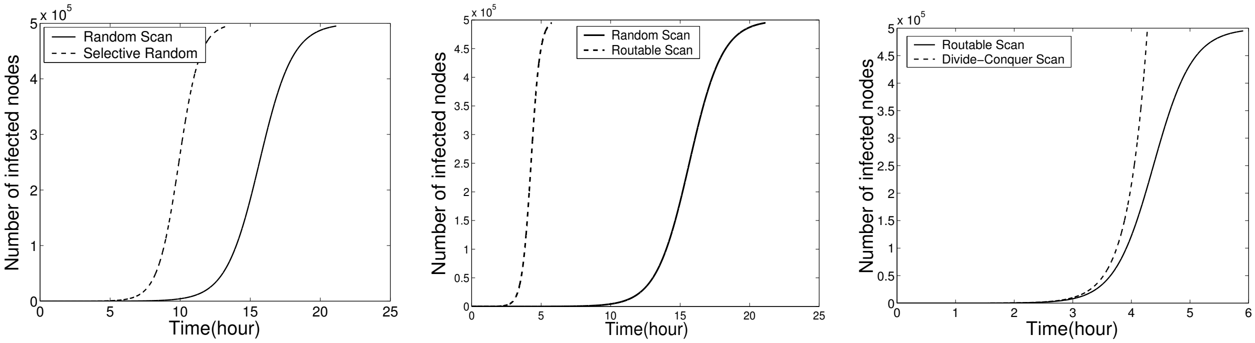
2.2 Routable Scan

In addition to the reduced scanning address space, if a worm also knows which of the addresses are routable or are in use, then it can spread faster and more effectively and can avoid detection. This type of scanning technique, where unassigned IP addresses which are not routable on the Internet are removed from the worm's database, is called routable scan. One problem with this type of scan method is that the code size of the worm has to be increased as it needs to carry a routable IP address database. The database cannot be too large as it leads to a long infection time resulting in a slow down of the worm propagation. In the next subsection we design and analyze a possible routable scanning technique.

2.2.1 Design and Analysis of Routable Scan

BGP is the global routing protocol that glues the independent networks together. From the global BGP routing tables, globally routable prefixes can be obtained. Some BGP routing tables are also available on the Internet. This data provides worm designers with methods that can enhance the performance of worms. To find out the applicability of using a BGP table as the database, we analyze a typical global BGP routing table with two objectives. The first is to find how many prefixes are globally routable. The second is to find the minimum size of the database a worm must carry. This analysis is done in 3 steps:

1. The BGP table we used is from one of the Route Views servers[21]. The routing table contains about 112K prefixes. By removing the redundant prefixes, 49K independent prefixes were obtained. In addition, those prefixes that are known to be unallocated were removed by checking the BGP table with IANA's



(a) Random scan and Selective Random scan (used by Slapper worm).

(b) Routable and random scan.

(c) Divide-Conquer and routable scan.

Figure 2. Spreading speed of selective random, routable and Divide-Conquer scan.

IPv4 address allocation map[4] (for example, prefix 39.0.0.0/8).

2. The continuous prefixes from above are merged into address segments. For example, the address segments of prefixes 3.0.0.0/8 and 4.0.0.0/8 can be merged into a new IP address segment: [3.0.0.0, 4.255.255.255].

3. Using a distance threshold, address segments close to each other are combined.

After step 2 described above, 17,918 independent address segments are obtained. The total number of IP addresses contained in those segments is 1.17×10^9. Comparing this with the total number of IPv4 addresses, 27.3% of the IPv4 addresses are found to be globally routable. This number is about 10 times larger than the number of IP addresses contained in the DNS tables (147M). The former is the upper bound of reachable IP addresses while the latter could be a lower bound.

In step 3, using a threshold of 65,536 (one /16 prefix away), the number of address segments is reduced to 1926. The total number of addresses contained in them is now 1.31×10^9 (12.6% higher).

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, it is found that the largest 20% segments contribute to over 90% of all the IP addresses. Therefore, with a 3K bytes database, a worm can still infect about 90% of the routable IP addresses.

2.2.2 Spreading Speed of the Routable Scan

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. Hence, $T \approx 10^9$. The other parameters are set to the same values as used in Figure 2(a).

Figure 2(b), which shows the spreading speed of routable scan and random scan, is obtained using Equation (1). While the Code-Red like worm that uses random scan spends about 24 hours to infect most vulnerable machines, the Code-Red like worm using routable scan only spends about 7 hours. Clearly, routable scan greatly increases the worm spreading speed.

2.3 Divide-Conquer Scan

Instead of scanning the complete database, an infected host divides its address database among its victims. We call this 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. Machine B can then scan the other half. Using Divide-Conquer scan, the code size of the worm can be further reduced, because each victim will scan a different and also a smaller address space. In addition, the scan traffic generated by the victims is also reduced and more difficult to detect.

One weak point of Divide-Conquer scan is "single point of failure". During worm propagation, if one infected machine is turned off or gets crashed, the database passed to it will be lost. The earlier this happens, the larger the impact. Several solutions exist that can be used to overcome this problem.

One possible solution is the generation of a hitlist, where a worm infects a large number of hosts before passing on the database. Another solution 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 solution is to randomly decide on whether to pass on the database or not.

2.3.1 Spreading Speed of Divide-Conquer scan

Consider a simple case of Divide-Conquer scan where the worm does not visit the addresses it has already visited. This reduces the probing range. We now have the following spread pattern for the Divide-Conquer scan:

$$n_{i+1} = (N - n_i)[1 - (1 - \frac{1}{10^9 - \sum_{j=0}^{i-1} n_j s})^{sn_i}] + n_i \quad (2)$$

where $i \geq 0$.

Using the same set of parameters used for the routable scan, Figure 2(c) shows that the Divide-Conquer scan is much faster than the routable scan, although the spreading process is more complicated. Furthermore, for the Divide-Conquer scan, even when there are few uninfected vulnerable machines, the spreading speed increases, rather than slowing down unlike the earlier cases.

2.4 Hybrid Scan

Limiting the scan targets by a specific address database might miss many vulnerable hosts that are not globally reachable. To solve this problem, the attacker can combine routable scan with random scan at a later stage of the propagation (when most addresses have been scanned) to infect more machines. This type of scanning technique is called Hybrid scan. The advantage of this technique is that even though the propagation has been detected, the hybrid scan can be used to infect more number of machines as it is already too late for effective defense.

Considering the fact that a large number of machines use private IP addresses and are hidden and protected by gateway machines from the Internet, better performance can be achieved if those addresses can be scanned with more power. In addition, each network interface card of a victim could be used for the worm scan to reach more subnets.

2.5 Extreme Scan Methods

In this section, we introduce and analyze several extreme scan methods. We call them "extreme" because the worms need to use some brute-force techniques to create a scan target address database. Because the worm scans only the hosts contained in the database, it can avoid from being detected. On the other hand, each method entails some critical limitations. That is, worms using extreme scan methods might spread very slowly or have weaknesses that can be exploited for detection.

2.5.1 DNS Scan

The worm designer could use the IP addresses obtained from DNS servers to build the target address database. Its advantage is that the gathered IP addresses are almost always definitely in use. However, it also has some problems. First, it is not easy to get the complete list of addresses that have DNS records. Second, the number of addresses is limited to those machines having a public domain name. From the observations of David Moore[6], we know that about half of the victims of Code Red I did not have DNS records. Third, because the worm programs need to carry a very large address database, the worms will spread very slowly.

Using the AAWP model, DNS scan's spreading speed can be analyzed. Here, $T \approx 10^8$. We assume the infection time t is roughly proportional to the worm's code size due to the transmission time. We use Code-Red I v2 as the standard, which has a code size of 5K bytes. Other parameters in Equation (1) are the same as used before. Figure 3(a) shows that DNS worm spread can barely start because of the huge address database the worm must carry.

2.5.2 Complete Scan

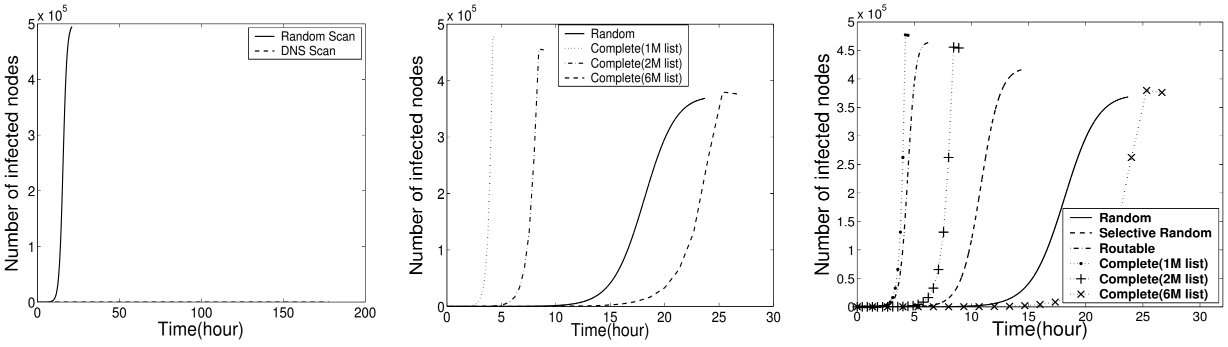
This is the most extreme method a worm designer might use to prevent the worm scan from being detected. The worm designer can use some method to get the complete list of assigned IP addresses and use the list as the target address database. Then, it is hard for a monitor to distinguish the malicious scans from the legitimate ones. The Flash Worm introduced by Staniford[12] is similar to this type of worm.

Using this method, the size of the attacker's code will also be very large because a large address list has to be carried with the code. From our former analysis, the number of distinct host addresses should be more than 100 million. Without compression, the list size will be at least 400M bytes, which will greatly slow down the infection time. However, for a specific vulnerability, the list could be reduced to a reasonable size. For example, if the objective of the attack is a security hole on routers, the attacker could collect a list of routers. As the number of routers on the Internet is much smaller, a complete scan will be applicable.

Using the same method and parameters used for the DNS scan, we can analyze the performance of complete scan. Figure 3(b) shows that when the address database is larger than 6M bytes and if no database splitting is used, the Code Red like worm using complete scan will spread slower than the one using random scan. We introduce a small death rate of 0.002% and a patch rate of 0.0002% in this case to clearly differentiate the various cases of complete scan. Besides the code size and the death rate, the remaining parameters are the ones used to simulate the Code Red worm. The infection rate graph, therefore, does not rise to the maximum possible value of 500,000.

2.6 Comparison of the Worm Scan Methods

The basic difference between the various scan methods lies in the selection of the address database. If the code size is large and the average bandwidth possessed



(a) DNS scan and random scan.

(b) Complete scan and random scan.

(c) Comparison of scan methods.

Figure 3. Extreme scan methods and comparison of major scan methods

by the infected host is given, then the time taken by a worm to infect a host is roughly determined by the speed of transmission of a copy of the worm program. Hence, we can see that the better the granularity of the database, the stealthier the scan technique. However, the size of the worm program will inevitably increase. There exists a tradeoff between the size of the address database and the speed of the worm spread. Figure 3(c) shows this tradeoff. Besides the list size and the death rate as in the case of the complete scan, other parameters in this figure are still the ones used to simulate the Code Red worm.

We can see that when the worm scan uses a better address database, similar to routable scan, the probability that a vulnerable machine gets infected is larger. For selective random scan and routable scan, the spread is faster than in the random scan. However, for a worm using complete scan, when the database size is larger than 1M bytes, the worm will spread slower than a worm using routable scan. A 1M bytes database can be translated to a list containing around 250K vulnerable hosts, which is not very large considering the number of Code Red victims. When the size of the list is larger than 6M bytes, the worm using complete scan will spread even slower than the worm using random scan. Although a worm using complete scan can effectively avoid being detected, the size of the address database cannot be too large. We can also see that slower spread will reduce the total number of victims the worm can infect.

We do not include Divide-Conquer scan here because we are trying find scan methods that are easy to implement. Considering the average power of the machines that can be used by the worm, we find that, a worm could use routable scan first and random scan later when most of the routable addresses have been infected. This combination does not cost much and leverages benefits from different scan methods.

3 Worm Detection

The main focus of this section is to detect worms using various scan techniques. Worm scan detection is raising an alarm upon sensing anomalies that are most likely caused by large scale worm spreads. Our goal is to quickly detect unknown worms on large enterprise networks or the Internet while making the false alarm probability as low as possible. In the following sections, we first present our generic worm detection architecture. We then present the design and analysis of a simple detection algorithm, called, *victim number based algorithm*.

3.1 Generic Worm Detection Architecture

To detect worms, we need to analyze Internet traffic. Monitoring traffic towards a single network is often not enough to find a worm attack. The traffic pattern could appear normal during a worm attack because the worm has not yet infected the network or will not infect it at all. Therefore, we have to monitor the network behavior at as many places as possible in order to reduce the chance of false alarms.

To achieve this, we need a distributed detection architecture. The architecture monitors the network behavior at different places. By gathering information from different networks, the detection center can determine the presence of a large scale worm attack. Problems such as where the monitors should be deployed, which addresses need to be monitored and how the detection system works need to be considered in designing the detection architecture.

We propose a generic traffic monitoring and worm detection architecture, as shown in Figure 4. The architecture is composed of a control center and a number of detection components. To reduce noise, traffic towards inactive addresses is preferred to be used for worm detection. The detection components will pre-analyze the traffic and send preliminary results or alarms to the control center. The control center collects reports from the detection components and makes the final decision on whether there is anything serious happening.

3.1.1 Implementation of Detection Components

The detection components can be deployed in one of the following two places.

1. Monitors In Local Networks: The detection components can be implemented on virtual machines or on the gateways of local networks.

2. Traffic Analyzers Beside Routers: Detection components can also be traffic analyzers beside the routers, observing the traffic of a set of addresses. Because of the processing costs associated with this form of detection, the analyzers cannot use complicated filtering rules. For example, they may only watch traffic sent towards unallocated /8 networks instead of particular addresses or they may just count the traffic volume.

3.1.2 Address Space Selection

The detection network consists of addresses monitored by detection components. Although a large detection network is highly desirable, the number of addresses the system can manage is limited. New worms might not scan the whole IP address space. In addition, different worms might use different target spaces. To make packet collection efficient and effective, we need to deploy the detection components so that the detection network overlaps as much as possible with the worm scan target space.

For the random scan, detection components can be deployed anywhere. For the routable scan or the Divide-Conquer scan, deploying detection components for allocated IP address spaces is better. Therefore, to detect worms that may be using any of the three scan methods, it would be better to use the latter strategy.

To collect inactive addresses, IP address blocks that are not assigned or known to be inactive for applications (for example most dial-up users do not have HTTP servers) are used. For the attackers using a random set of such address blocks, it will be difficult to know the detection network. Some virtual machines could also be setup to respond to the scan packets they receive. Hence, the attacker will not be able to tell the difference between inactive addresses and active addresses.

3.2 Victim Number Based Algorithm

Using the detection architecture, we need to design algorithms to detect anomalies caused by worms. Since a new worm's signature is not known beforehand, a small number of packets is not enough to detect the worm. It is abnormal to find a large amount of scan traffic sent towards inactive addresses. This is, however, prone to false alarms because the scan traffic can be caused by other reasons (such as DDOS and software errors). Therefore, it is necessary to find some unique and common characteristics of worms.

Serious worm incidents usually involve a large number of hosts that scan specific ports on a set of addresses. Many of these addresses are inactive. If we detect a large number of distinct addresses scanning the inactive ports, within a short period of time, then it is highly possible that a worm attack is going on. We define the addresses from which a packet is sent to an inactive address as *victims*. If the detection system can track the number of victims, then the detection system has a better performance. Hence, a good decision rule to determine if a host is a victim is necessary.

3.2.1 Victim Decision Rules

To decide if a host is a victim, the simplest decision rule is that at least one scan packet is received by an inactive host. We call this the "One Scan Decision Rule" (OSDR). Using this rule, the number of victims detected by the detection system over a period of time can be modelled as:

$$V_k = \sum_{i=0}^{k} [n_i - n_{i-1}][1 - (1 - D/T)^{(k-i)s}] \qquad (3)$$

where V_k is the number of victims detected by the system up to time tick k (here the death rate and patch rate are both assumed to be zero and $n_{-1} = 0$) and D is the detection network size. In [20], Chen et. al. proved that for an address space containing more than 2^{20} addresses, the number of victims determined by the detection system closely matches the dynamics of the random scan's propagation.

Though simple, OSDR is susceptible to daily scan noise. Any host that has a scan packet collected by the detection network is considered to be a victim. David Moore's work[6] showed that "two scans captured by the host leads to a victim". This "Two Scan Decision Rule" (TSDR) works well with noise and reflects the incessant feature of worm scans. We focus on TSDR in this paper.

TSDR reduces most of the sporadic scans caused by port scans or software errors. Apparently, higher the number of scan packets used to make the decision, higher is the decision accuracy. Yet this allows more victims to slip away from the detection system. The number of victims detected by a detection system using TSDR up to time tick k can be calculated by the equation:

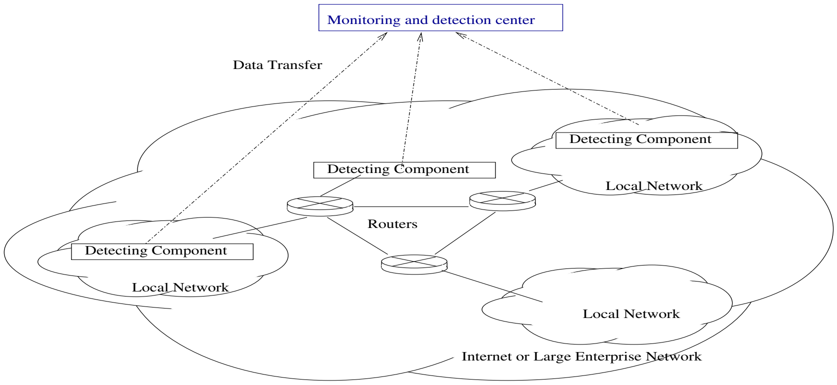


Figure 4. A generic traffic monitoring and worm detection architecture.

$$V_k = \sum_{i=0}^{k}[n_i - n_{i-1}][1-\rho^{(k-i)s} - \rho^{(k-i)s-1}(1-\rho)(k-i)s] \tag{4}$$

where the number of infected hosts up to time tick i is n_i and $\rho = (1-\frac{D}{T})$. Here, we assume the death rate and patch rate to be zero and $n_{-1} = 0$. On the right hand side of the equation, $n_i - n_{i-1}$, is the number of newly infected machines during time tick i. The total number of machines detected at the end of time tick k is the sum of machines detected at every time tick i before k. $\rho^{(k-i)s}$ denotes the fraction of victims infected during time tick i but were never detected up to time tick k. The term $\rho^{(k-i)s-1}(1-\rho)(k-i)$ denotes the fraction of victims infected during time tick i but only one of their scan packets has been detected by time tick k.

Equations (3) and (4) above are based on the assumption that each new victim will use the same address space as the destination address space used by the worm. This is a common feature of the two scan methods presented earlier. We also assume that the address space used for detection is always a subset of the address space used by the worm. The death rate and the patch rate are ignored.

For the Divide-Conquer scan, each victim will scan the addresses only within its piece of the target space. Therefore, the fraction of victims captured by the detection system is the fraction of address space that the detection system covers. If we assume that the victims are randomly distributed around the address space used by the attacker, and $s \geq 1$, then the number of victims detected by time tick k can be expressed by a much simpler equation:

$$V_k = \frac{D}{T} n_{k-1} \tag{5}$$

3.2.2 Detection Threshold

The victim number based algorithm consists of three parts: First, all scan packets with inactive destination addresses are gathered. This task is accomplished by the detection architecture. Second, the victims are retrieved from the gathered addresses. This has been discussed in the previous subsection. Third, an adaptive threshold is needed to determine when a surge is large enough. During the monitoring of blocks of inactive addresses, if there is a quick surge in the number of distinct victims, then there is an evidence of a serious worm attack. The victim number based detection algorithm is based on the above principle.

Using only inactive addresses, the noise caused by normal scan traffic and other attacks can be reduced. This method requires simple filtering rules to gather suspicious packets and count them. Hence, it is very fast. The following facts can also be used in order to demonstrate the effectiveness of our solution. Port scans are usually done by a limited number of hosts and normal DOS or DDOS traffic will not focus on inactive addresses. Hence, these will not cause much impact on our solution.

To combat noise that might be received by the architecture, the scan volume is kept stable. This allows us to use an adaptive threshold that can help in detecting abnormality. If there is deviation in the number of victims obtained within a certain period of time, then the victim number based algorithm's threshold should be able to reflect it. The threshold could be expressed by:

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

$$E[V_i] = \frac{1}{k}\sum_{j=i-k}^{i-1} V_j \tag{7}$$

where T_i is the threshold to be used by the system at time tick i; γ is a constant value called threshold ratio; V_j is the number of new victims detected during each k time ticks starting from $i-k$ time ticks; k is the learning time of the system, which is the time taken by the system to calculate $E[V_i]$; $E[V_i]$ is the rate of increase in the number of new victims at every time tick i averaged over the k time ticks. If the rate of increase of victims detected during the i^{th} time tick is greater than the threshold value set during that time, then there is abnormality present in the system. In practice, we also need to find continuous number of such abnormalities to determine worm activity. The number of continuous abnormalities needed to detect worm activity is denoted as r. A tradeoff is necessary in selecting the value of r. A large value of r give a better detection performance but take longer time whereas a smaller value might result in false positives but take lesser time.

Initially, the number of new victims detected is large as there are only a small number of addresses in the database. This leads to a large rate of increase showing peaks in the detection curve. However, as the database gets larger and larger the rate of increase of new victims observed decreases. In order to smooth these peaks of the initial learning process, we need to have some way for entries to expire in the database. A simple method is to use new databases everyday. For a specific day, 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 dead and removed from the victim list. If a scan packet is received from the victim before L expires, its life time is then reset to L. Using this method, the size of the database can be kept stable. However, tracking the time for each address is expensive. We use the first method for our solution.

Another static threshold that reflects a maximum possible victim number observed by the system over a period can also be used. Whenever this threshold is reached, an alarm is raised regardless of the former self-learning threshold.

It needs to be pointed out that when the worm scan level is comparable to the normal noise level, the worm will not be detected. Therefore, the system will usually raise an alarm only when it detects large scale worm attacks. We set the parameters γ, r and k using the technique shown in Appendix B.

3.3 Traffic Trace Used To Validate The Algorithm

Once the parameters γ, r and k are appropriately selected (Appendix B), we need to test the solution with real traffic traces obtained from worm incidents in order to validate the detection method. However, it is very difficult to find such Internet traffic traces which are publicly available. As background traffic, we use Internet traffic traces from the WAND research group[16]. The traffic traces are gathered from the gateways on the campus network at University of Auckland, New Zealand, who own a /16 prefix. Using worm infection dynamics obtained from the AAWP model and Equation (4), we simulate the number of worm victims detected by a /16 network over time. We find that for the case of random scan and routable scan the infection dynamics can be easily captured using a /16 network (Appendix A). 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. From these we verify our detection solution.

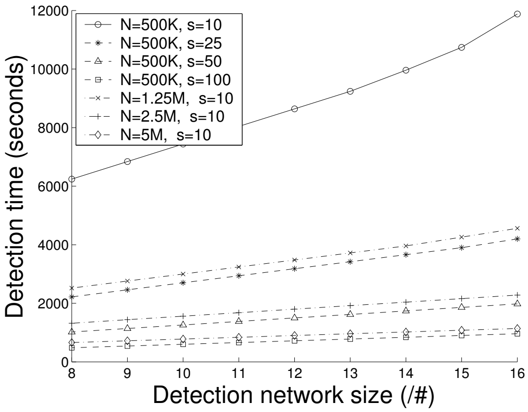
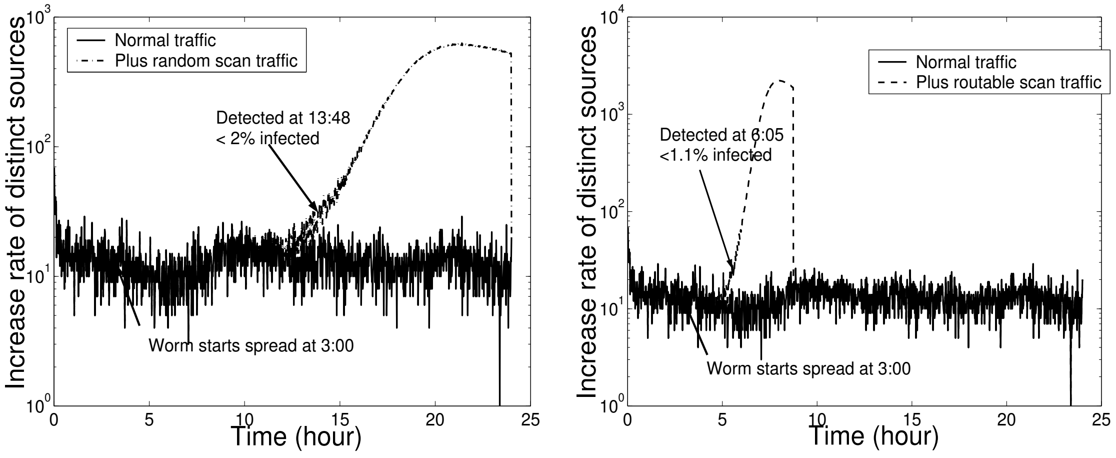


Figure 5. Performance of victim number based algorithm for various sizes of detection network

We use 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), it is good to simulate the Code Red worm traffic. Hence, the packets we are interested in are the SYN packets sent towards TCP port 80 (HTTP). In the simulation, we assume that the worm starts at 3 o'clock in the morning.

3.4 Validation of Victim Number Based Algorithm

Before we present the actual validation, the algorithm is tested for different number of vulnerable machines and different possible scan rates. Figure 5 shows the change in detection time with the detection network size as a function of total number of vulnerable machines (N) and scan rate (s). We assume the noise level is 1000 (from [15]. This is because, the total number of distinct scan sources is 5000 every day, so we assume that for an important port, the average number of the scan sources is 1000. The γ value we use is 3. The uppermost curve in the 5 is for the detection of a Code-Red like worm with 500,000 vulnerable machines. Note that as the detection network size increases, the number of machines decreases (/8 network has more machines than /16). The other curves show the change in detection time for various values of N and s. From the plot, we can find that when the worm attack is more serious (more machines are vulnerable or scanning becomes faster - the most important factors



(a) Variation of victim number in case of random scan

(b) Variation of victim number in case of routable scan

Figure 6. Validation of algorithm with Internet traffic traces from the WAND research group.

in the worm spread),the detection time will be shorter. Therefore, for worms more serious than the Code-Red, and a detection network size smaller than /12, our solution could detect the worm within 2 hours when less than 1% of vulnerable machines are infected.

Figure 6(a) is plotted from the simulated Code Red worm dynamics (that start at 3 o'clock in the morning). It shows that with the /16 network (Appendix A), there is a rapid increase in the victim number during the Code Red worm attack. Figure 6(b) shows the same worm attack using routable scan. Using a threshold with $\gamma = 3$, $k = 200$ minutes and $r = 10$ (obtained from Appendix B) in Equation (7), the worm attack can be detected within 11 hours after the Code-Red worm begins to spread. At this time, less than 2% of the 500,000 vulnerable machines are infected.

We already know that the faster the worm spreads, the faster it can be detected. Now, if the Code Red worm uses routable scan, the worm can be detected within 3 hours. At this time, less than 1.1% of the total number of vulnerable machines are infected.

From the Figure 7, we see that if the Code Red worm uses Divide-Conquer scan, the detection network will not be able to detect it (Appendix A). Notice that the scale of the X axis have been changed to clearly depict the inability to detect the divide conquer scan. This is because the /16 network is too small and on average can only detect $\frac{1}{2^{16}}$ of the victims. In the Code Red worm's case, this is less than 10 victims. However, we expect that by using larger detection networks containing only inactive addresses, the Divide-Conquer worm scan can also be detected.

Besides the Code Red and routable worm, we also want to answer a more important question - to what extent does the victim number based detection algorithm work for other worms with different scanning rates, destination address space sizes and total number of vulnerable machines. Figure 8(a) shows the detection performance (fraction of vulnerable machines got infected upon detection) of the victim number based detection for worms with varying scan rates and destination address space sizes with total vulnerable machine number as 500,000. Here, $k = 200$, $r = 10$, $\gamma = 3$ (from Appendix B). From the plot, we can see that the detection performance improves when the destination address space size decreases and the scan rate is higher. For scan rates ranging from 2 scans per second to 100 scans per second and the destination address space size ranging from 1.3×10^9 to 2^{32}, the worm can be detected before 1.5% of the vulnerable machines are infected.

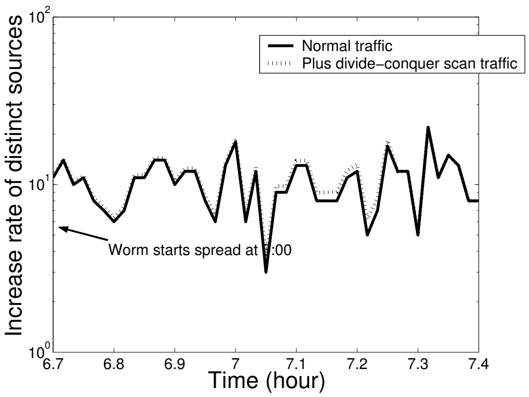
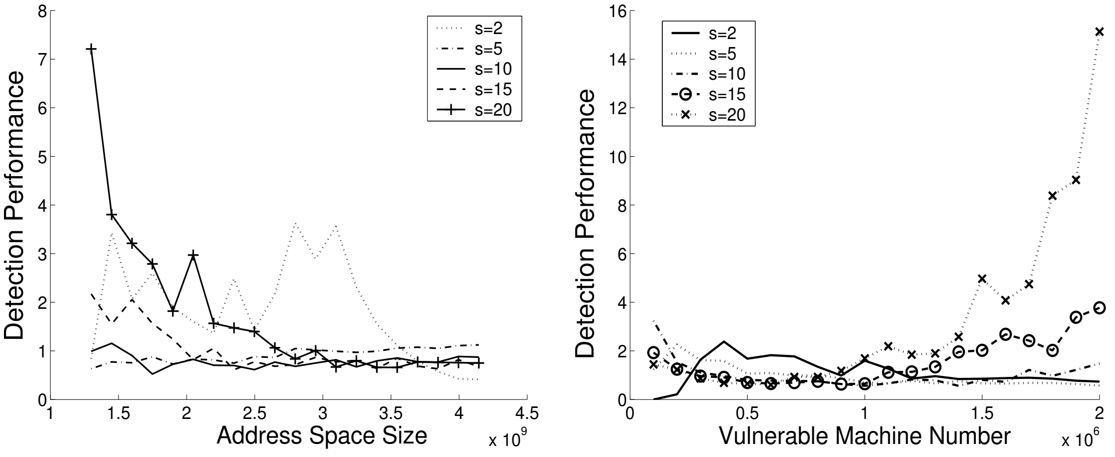


Figure 7. Variation of victim number for divide-conquer scan traffic

Figure 8(b) shows the detection performance of the victim number based detection for worms with vary-



(a) Detection performance Vs. Destination address space size as a function of scan rate.

(b) Detection performance Vs. number of vulnerable machines as a function of scan rate.

Figure 8. Detection performance.

ing scan rates and vulnerable machine number with destination address space size as 2^{32}. Here, $k = 200$, $r = 10$, $\gamma = 3$. From the plot, we can see that the detection performance improves when the vulnerable machine number increases and the scan rate is higher. For the scan rates ranging from 2 scans per second to 20 scans per second and the vulnerable machine number ranging from 100,000 to 2,000,000, the worm can be detected before 4% of the vulnerable machines are infected. We can see from the plot that for scan rates higher than 20 scans per second and for vulnerable machine numbers lower than 400,000, the detection performance decreases because the worm spreading is slow and can be detected by a network as small as /16. We found that when the vulnerable machine number reaches around 2,000,000, the detection performance degrades. This is because the worms spread so fast that when we detect it in only 1 minute or a few minutes after the worm starts to spread, the fraction of vulnerable machines that are infected is already high. Therefore, in the plot, when vulnerable machine number is higher than 2,000,000 and when the scan rate is higher than 20 we can still detect the worm in a short time although the benefit of the detection could be much less.

4 Conclusions and Future Work

When the attackers are more sophisticated, probing is fundamentally not a costly process. From the discussions above, it seems that the game would favor the attackers when the Internet links are fast enough and the size of the code is not critical to the propagation speed.

This does not imply that monitoring is of no use. In future, an efficient traffic monitoring infrastructure will be an important part of the global intrusion detection systems. A consequence of the worm detection method is that the attackers will have to use a limited number of IP addresses to scan the Internet. Therefore, the impact of worm scanning on the Internet traffic will be reduced.

In this paper, we discussed different types of scan methods and their effects on future worm propagation. We find that as the backbone link speeds and hosts of greater capacity are affordable to the attackers, it will be more difficult for us to detect worm scanning from the Internet traffic. However, the detection methods can still be useful in that it forces the attacker to use lesser traffic and scan more slowly and cautiously.

We designed two new scan techniques, routable scan and Divide-Conquer scan. Basically, they both use the idea of a routable IP address list as the destination base where the scan object is selected. A routable worm is easy to implement; it poses a big menace to the network security. We must keep in mind that anytime in future the next worm incident may be worse.

For this strain of scan methods we designed a detection architecture. Specific detection methods were also designed. We find that using the victim number based algorithm, with a /16 network, worms more serious than the Code Red and the Slammer can be detected when less than 4% of vulnerable machines are infected.

We observed that the number of false alarms increase in the case of a DDoS attack or in the case of a hot website visit. Future work lies in this direction of developing an integrated approach to further improve the above proposed technique and develop an efficient mechanism to fight worm attacks.

References

[1] D. Moore, *The Spread of the Code-Red Worm (CRV2)*, http://www.caida.org/analysis/ security/ code-red/coderedv2/analysis.xml

[2] Fyodor, *The Art of Port Scanning*, http://www.insecure.org, Sept. 1997

[3] J. O. Kephart and S. R. White. *Measuring and Modeling Computer Virus P revalence*, Proc. of the 1993 IEEE Computer Society Symposium on Research in Security and Privacy, 2-15, May. 1993,

[4] *Internet Protocol V4 Address Space*, http://www.iana.org/assignments/ipv4-address-space/

[5] T. Liston. *LaBrea*, http://www.hackbusters.net/ LaBrea/

[6] D. Moore. *Code-Red: A Case Study on the Spread and Victims of an Internet Worm*, http://www.icir.org /vern/imw-2002/imw2002-papers/209.ps.gz

[7] 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

[8] R. Thomas, *Bogon List v1.5 07 Aug 2002* http://www.cymru.com/Documents/bogon-list.html

[9] Internet Storm Center, *OpenSSL Vulnerabilities*, http://isc.incidents.org/analysis.html?id=167, Sept. 2002,

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

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

[12] 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/

[13] S. Cheung. *Graph-based Intrusion Detection System (GrIDS)* http://seclab.cs.ucdavis .edu/arpa/grids/PI_meeting _(Savannah).pdf, Jan. 1999

[14] T. Liston, *Welcome To My Tarpit - The Tactical and Strategic Use of LaBrea*, http://www.hack busters.net/

[15] V. Yegneswaran, P. Barford and J. Ullrich *Internet Intrusions: Global Characteristics and Prevalence*, SIGMETRICS 2003.

[16] *Waikato Internet Traffic Storage*, http://wand.cs .waikato.ac.nz/wand/wits/index.html

[17] *Warhol Worms: The potential for Very Fast Internet Plagues*, http://www.cs.berkeley.edu/ nweaver/warhol.html

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

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

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

[21] *University of Oregon Route Views Project*, http://www.routeviews.org

[22] *Global Slapper Worm Information Center*, http://www.f-secure.com/v-descs/slapper.shtml

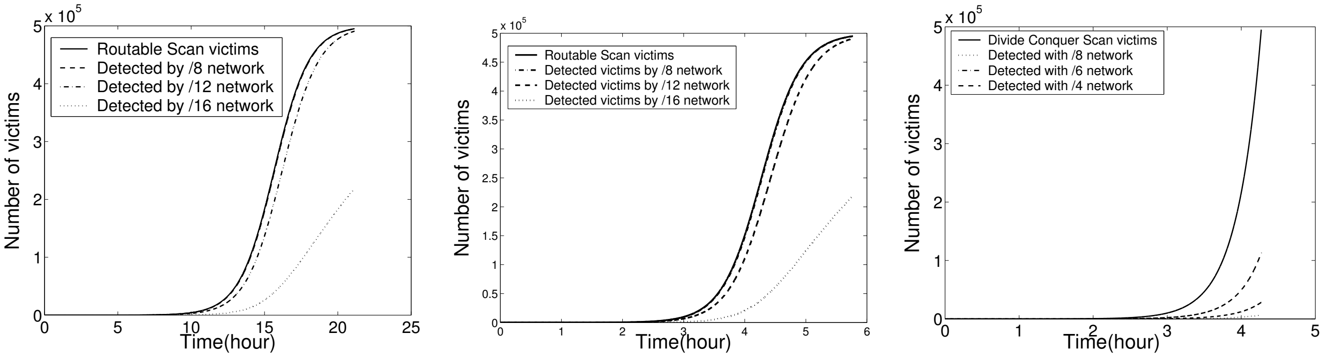
[23] C. Zou, L. Gao, W. Gong and D. Towsley, *Monitoring and Early Warning for Internet Worms*, Umass ECE Technical Report TR-CSE-03-01, 2003.

5 Appendices

A Impact of Network Size on Observed Number of Victims

As our solution uses only the inactive addresses, it applies to worms that blindly probe an address space. This is because they do not know which hosts are vulnerable beforehand. This is a reasonable assumption, otherwise a worm that knows the list of vulnerable hosts, could just infect them one by one. In this case, the weak worm scan traffic can go completely undetected. An ideal detection system should with a reasonable sized detection network be able to detect large scale worm propagation soon after the victim number becomes much larger than the normal scan noise level.

Using the formulas derived earlier, we can analyze the performance of our solution for worms using different scan methods. We define performance as the least size of detection network needed to ensure that a worm is detected within a certain time. Intuitively, the larger the detection network, the faster will the number of victims detected approaches the real victim number and thus exceeds the threshold. Below are the evaluations for worms using several scan methods. We use TSDR in order to detect scanning for each of the scan methods.



(a) Random Scan.

(b) Routable scan.

(c) Divide-Conquer scan.

Figure 9. Impact of detection network size on observed number of victims

1. Random Scan: Figure 9(a) shows the detection curve of for random scan. We see that the detection curve approaches the infection curve when the detection network size is over 2^{20} (a /12 network).

2. Routable Scan: For routable scan, Equation (4) can be used to simulate the detection curve. From Figure 9(b), we see that the detection curve also catches up closely with the infection curve when the detection network size is over 2^{20}.

3. Divide-Conquer Scan: From Figure 9(c), we can see that even when a /4 detection network is used for Divide-Conquer scan, the detection curve still lags far behind the infection curve (in the /4 case, the fraction of victims that could be detected is less than 1/16). This means that Divide-Conquer scan is difficult to detect.

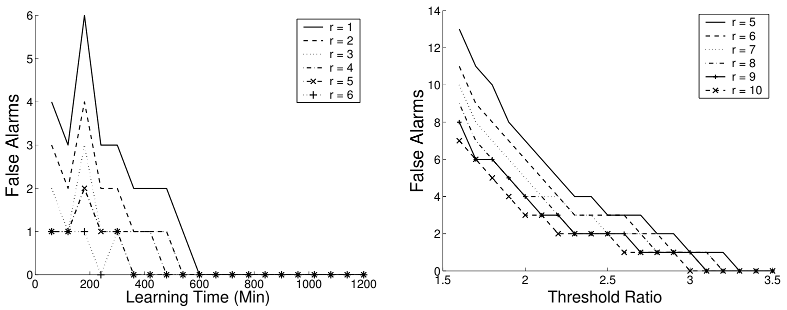
From above, we can observe that infection dynamics of the first two scan methods can be captured with same performance. The Divide-Conquer scan is very hard to detect as only a fixed fraction of victims can be detected. This is because random scan and routable scan victims share the same target address space while the Divide-Conquer victims use only a piece of the whole address space. When the detection network size or the vulnerable machine number is too small, our detection solution might fail. Fortunately, Divide-Conquer scan has its limitations and can be combined with random scan to enhance its performance making our solution to work in that particular case also.

B Selection of Parameters

In selecting parameters for the detection system the most important factor we consider is false alarms. By applying our detection system for various parameter settings we were able to choose the parameters with reasonable efficiency. The learning time, k, cannot be too long as it can lead to a lower detection performance by making the detection time longer. This can in turn lead to increased number of infections. We choose a value of 200 minutes for k. We also find that a value of $r \geq 6$ leads to a decreased number of false alarms. A value of $\gamma \geq 3.4$ makes worm detection difficult. Figures 10(a), 10(b), 11(a) and 11(b) justify the above assumptions. We assume that the June 12th traffic trace does not contain major worm attack traffic in obtaining the previous plots. Figure 10(a) shows that when $\gamma = 3$, by selecting $r \geq 6$ and using a learning time $k \geq 200$ minutes, we can eliminate false positives.

In Figure 10(b), we can see that even when using r as large as 10, we still need γ to be larger than 3 to eliminate false positives completely. From Figure 11(a), we can see that when γ is 3, k is around 200 and $r \geq 10$, the detection time is the lowest. One thing we can observe from 11(b) is that when γ is larger than 3.4, the worm cannot be detected.

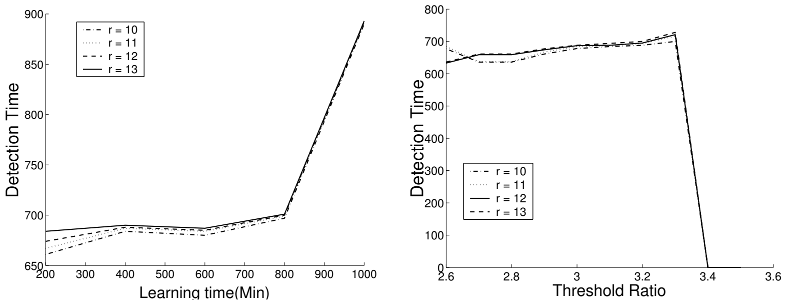
The verification of the above parameters for the routable scan method is shown in Figures 12(a) and 12(b). For the verification of the parameters, before the worm is detected, because the time is very short, there are no false alarms at all. For the detection performance, Figure 12(a) and Figure 12(b) show similar results as in the Code Red worm's case.



(a) With a fixed threshold ratio ($\gamma = 3$)

(b) With a fixed learning time ($k = 200$)

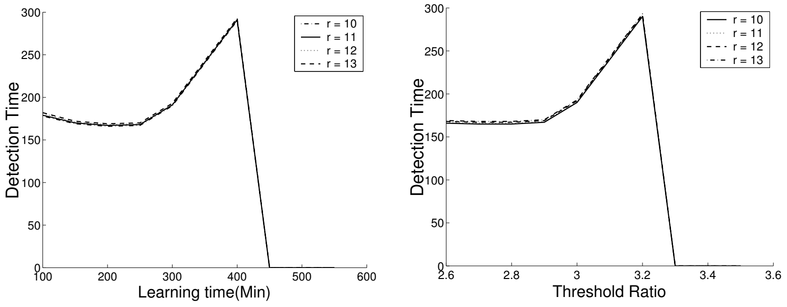
Figure 10. Selection of parameters with normal traffic. 'r' is the number of continuous abnormalities observed in order to take a decision.



(a) With a fixed threshold ratio ($\gamma = 3$)

(b) With a fixed learning time ($k = 200$)

Figure 11. Verification of parameters with Code Red worm like traffic. 'r' is the number of continuous abnormalities observed in order to take a decision.



(a) With a fixed threshold ratio ($\gamma = 3$)

(b) With a fixed learning time ($k = 200$)

Figure 12. Verification of parameters with routable worm traffic. 'r' is the number of continuous abnormalities observed in order to take a decision.