Classification of Different Computer Worms with Dynamic Detection Using Victim Number Based Algorithm

Ravinder Nellutla Asst. Prof.[1], Vishnu Prasad Goranthala Assoc. Prof.[2] Fasi Ahmed Parvez Assoc.Prof.[3]

[1](Department of Information Technology, Kamala Institute of Technology and Science, Singapur, Huzurabad, Karimnagar.

[2](Department of Computer Science and Engineering / Information Technology, Balaji Institute of Engineering Sciences, Laknepally, Narsampet, Warangal.

[3](Department of Computer Science and Engineering / Information Technology, Balaji Institute of Engineering Sciences, Laknepally, Narsampet, Warangal.

ABSTRACT

The Internet has developed to give many benefits to mankind. The access to information being one of the most important. Worms cause major security threats to the Internet. Worms are software components that are capable of infecting a computer and then using that computer to infect another computer. The cycle is repeated, and the population of worm-infected computers grows rapidly. Smart worms cause most important security threats to the Internet. The ability of smart worms spread in an automated fashion and can flood the internet in a very short time. 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 worm scan 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. 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 analyzed different scan methods and compared them, we find that the victim number based algorithm can dramatically increase the spreading of speed of worms.

Key terms: Worms, Network security, Random Scan, Virus, Victim Number Based Algorithm,

I. Introduction

Worms are one of the most ill defined concepts in Network Security. There is still no universal consensus on the definition of the worm. Usually worms and viruses display similar characteristics and their intention is also similar. To define worms, we will use the following points and then define worm based on these points.

The propagation of the worm is based on exploiting vulnerabilities of computers on the Internet. Many real-world worms have caused notable damage on the Internet. These worms include “Code-Red” worm in 2001 [1], “Slammer” worm in 2003 [2], and “Witty”/ “Sasser” worms in 2004 [3]. Many active worms are used to infect a large number of computers and recruit them as bots or zombies, which are networked together to form botnets [4].

Worms can start on a host (Computer) in various fashions. It may be an attachment to a mail and when the attachment is opened, will execute the code written in the worm. This is called "invocation by human intervention". It may also start without any human intervention. For example, rebooting the system. It affects the host. In contrast to computer viruses, it can affect anything on the host. It may corrupt the files on the host. It may affect communication of the host with other systems. It may disable the anti-virus software on the host, which will enable it to cause more damage. Computer Viruses in the other hand are very specific to files. Worms have a broader scope of attack than viruses. Worms are self replicating codes. This is the most distinct feature of a worm. Once they infect a host, they will try to find a nearby host which they can access, and copy themselves to that host. There it will perform the same actions that it performed on the original host.

"A worm is a computer program, which can self-replicate and propagate over the network, with or without human intervention, and has malicious intent."

1.1. Differences between virus and worms:

VIRUS	WORM
A Virus is a program that is designed to spread from file to file on a single Pc.	A worm is designed to copy itself (intentionally move) from PC to PC, via networks, internet etc.
It does not intentionally try to move to another PC.	A worm does not need a host file to move from system to system, where as a virus does.
It must replicate and execute itself to be defined as a virus	Worms spread more rapidly than viruses.

II. RELATED WORK

1.1. Active worms:

Active worms are similar to biological viruses in terms of their infectous and self-propagating nature. They identify vulnerable computers, infect them and the worm-infected computers propagate the infection further to other vulnerable computers. In order to understand worm behavior, we first need to model it. With this understanding, effective detection and defense schemes could be developed to mitigate the impact of the worms. For this reason, tremendous research effort has focused on this area.

Active worms use various scan mechanisms to propagate themselves efficiently. The basic form of active worms can be categorized as having the Pure Random Scan (PRS) nature. In the PRS form, a worm-infected computer continuously scans a set of random Internet IP addresses to find new vulnerable computers. Other worms propagate themselves more effectively than PRS worms using various methods, e.g., network port scanning, email, file sharing, Peer-to-Peer (P2P) networks, and Instant Messaging (IM) [7], [8]. In addition, worms use different scan strategies during different stages of propagation. In order to increase propagation efficiency, they use a local network or hitlist to infect previously identified vulnerable computers at the initial stage of propagation [13], [14]. They may also use DNS, network topology, and routing information to identify active computers instead of randomly scanning IP addresses [10], [11]. They split the target IP address space during propagation in order to avoid duplicate scans [10]. Li et al. [12] studied a divide-conquer scanning technique that could potentially spread faster and stealthier than a traditional random-scanning worm. Ha and Ngo [5] formulated the problem of finding a fast and resilient propagation topology and propagation schedule for Flash worms. Yang et al. [6] studied the worm propagation over the sensor networks.

III. 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.

I. Popular Worms

3.1. Creeper Worm

Released in early 1970's and written by Bob Thomas, it was an experimental program to demonstrate the power of programming. Most of the worms written at the time were a result of fascination for self replicating programs by the programmers. There was not malicious intent and the worms did not hide. They were sent in clear. The Creeper worm was written to infect DEC PDP-10 computers running the TENEX operating system. The program used the ARPANET to propagate from node to node and display a message "I'm the creeper, catch me if you can!" A program, Reaper, was written to counter Creeper.

3.2. Morris Worm

Released in 1988 and authored by Robert Tappen Morris, was the first known worm that had malicious intent. According to the author, the worm was not suppose to cause any damage and was intended to gauge the size of the internet. It however, did cause DoS attacks. The worm exploited the vulnerabilities of Unix sendmail, rsh/rexec and weak passwords. The worm initiated a process on the host and found new hosts to propagate the code. Once it found a new host it would copy itself to the new host and start an additional process there. The worm has a condition to check if the worm is already running on the host. But Morris has programmed in such a way that the worm propagated to the new host even if the answer was "Yes". Every new instance of the worm on the host caused an additional process to be launched. And each new process slowed the system down until the system was unusable. The Morris worm is also considered as the Great Worm as it was first of its kind and it demonstrated the amount of impact such programs can have if they are not secured. It also changed the perception of system Downtime and Internet Security forever.

3.3. Melissa Worm

This was a worm that caused wide spread damage to the internet and for the first time huge losses to everyone around the planet. It caused over 400 million USD in damages across the globe and shutdown many organizations. It was written as a MACRO on Microsoft Word Document and this helped its widespread propagation. It was released in Mid March 1999 and was authored by David L. Smith. The worm was very simple in its concept, but demonstrated a new technique to propagate. Many of the worms that were written in the years to come, were derived from this concept in one way or another. The worm was present in the MACRO of a MS-WORD document and propagated as a document that supposedly contained passwords for 80 pornographic sites. If the user opened this document, and many of them did, it would execute the MACRO. Once the MACRO was executed, it would pick up the first 50 contacts from the users address book and mail a copy of itself to all the addresses. Since the worm was essentially an email worm and it mailed 50 address every time it infected a new host, many mail servers were clogged with the mails. This caused a wide spread DoS attack. Most of the techniques used by this worm laid the foundation or methodology for many variants and newer worms. Papa and Syndicate are two such variants.

3.4. Explore Zip

This worm took the concept of Melissa worm one step further. Melissa worm was not designed to reside on the system. ExploreZip was. The worm propagated via email, just like Melissa, and was present in an attachment called ZIPPED_FILES.exe. Once the user opened the attachment, the worm would seem like a self extracting zip archive and then error out. Behind the scenes it would install itself on to the system and register itself in the Windows Registry. The worm would then stay dormant and do nothing. When the user reboots the system, the worm would get activated and mail a copy of itself to all the people in the address book of the user on the host. It would also delete all the C and C++ source files from the hard drive. There is no record of the amount of damage done by this worm. Since all the computers are not started at the same time, it is unlikely that this worm could have caused any DoS attack. It was not instantaneous like Melissa.

3.5. I Love You

This was the first worm to take the cost of damage to billions of USD. An estimated damage caused by this worm was between 5 and 10 billion USD. The worm was written in VB Script and propagated as an attachment in the email with a message "ILOVEYOU". When users opened this attachment, it would register itself onto the Windows Registry. This would activate the worm after every restart of the system. It would then, search all the drives connected to the host for all files with extensions *.JPG, *.JPEG, *.VBS, *.VBE, *.JS, *.JSE, *.CSS, *.WSH, *.SCT, *.DOC *.HTA, *.MP3, *.MP2 and rename them to .VBS. It also had a component called WIN- BUGSFIX.EXE" or "Microsoftv25.exe". This was a password stealing program. The worm propagated across the network by using the addresses present in the address book of the user. Since the worm activated immediately and also on restart of the PC, the amount of email it generated crippled many mail servers and also individual PCs. The worm was allegedly authored by Irene, Onel de Guzman and Reomel Lamores from Philipines.

3.6. Code Red

This worm took the approach to attacking in a completely different direction. Instead of relying on mails address in the user's contact list, it performed network scanning and used the IP addresses connected to the host as a vector for propagation. It attacked the IIS servers and defaced many websites. It used the vulnerability of buffer overflows on IIS servers to execute binary code on the hosts. The initial worm did not check if the new host has windows or was running IIS. It also did not check if the IP address it was trying to access exists. The later versions of this worm were more inclined towards the local subnet rather than accessing some random IP. The total cost of damage was about 1.2 billion USD. It demonstrated a new technique or worm propagation.

3.7. Nimda

This was the next generation worm in its own league. It had 4 different propagation vectors. It could propagate via Websites, LAN, Emails and as executables. In emails it was disguised as a BASE-64(Binary) file readme.exe in the MIME Section. It would pick up the address retrieved from the user's MAPI Service. In the browser mode of propagation, the worm would rename many of the system files to .html and .asp. These pages would get executed and download the worm onto the machine, thus infecting the host. In the LAN Mode, it would copy itself on to all the writable shared directories that it could find. If the remote user opened these shared drives and if the "auto preview" option was enabled, the worm would infect the remote computer. It would them repeat the same process on the remote PC. The estimated cost of damage of this worm was about 8.75 billion USD.

3.8. Mydoom

This was the most notorious worms of all times with the highest damage of 22 billion USD. It propagated as a "Sending Failed" mail from the mail server and asked the user to click on the attachment to resent the mail. If the user opened the attachment, it would show that it's resending the mail and in parallel, installed the Understanding Worms. The worm would then send a copy of itself to all the address in the address book and also copy itself to Peer-to-Peer shared drives. The worm also opened a back door for the hacker to get back anytime .

3.9. Sasser

This worm was unique in the manner in which it was developed. The worm was reverse engineered from one of the patches provided for Microsoft Windows. The worm would exploit the vulnerability the patch was suppose to address and was targeted at systems that had not installed the update yet. It did not portray any new technological advance from the way the worm behaved. But the design of the worm was a step further in worm innovation. It targeted the LSASS component that represents Buffer Overflow and executed binary code on the hosts. Since buffer overflow causes erratic behavior or shutdown of the system, many organizations across the globe went down almost instantaneously. It caused a damage of over 14 billion USD and was authored by Sven Jaschan.

I. Worm Characteristics

Worms can be categorized by their target discovery technique, propagation carrier and distribution mechanism, activation and payload [8].

4.1 Target Discovery

Target discovery is the first step of the worm propagation, the purpose being to detect new hosts to infect. There are several possible techniques by which a vulnerable target can be discovered: by scanning, by use of various target lists and by passive monitoring [15]. Many of the most effective worms combine several of these techniques in order to use the best from
each technique.

4.1.1.Scanning

The scanning technique involves probing a set of addresses in order to detect vulnerable hosts. The simplest forms of scanning are sequential and random scanning. The former implies probing addresses sequentially from an address block, while the latter implies trying addresses from an address block in a pseudo-random fashion. Their simplicity makes them frequently used. To increase the efficiency of the target discovery mechanism, worm authors have suggested several optimizations for scanning worms. One optimization is the preference for local addresses in order to reduce latency. This is commonly referred to as island hopping because the worm's spreading pattern tends to resemble islands. In addition to reducing latency, island hopping will also reduce the number of encounters, and thereby possible detections and failed infection attempts, with firewalls and NATs. At the same time, it makes the worm more vulnerable in its initial stage, as total containment is possible if the worm is detected and isolated while still infecting hosts in the initial local network [15]. Another optimization is a bandwidth-limited scanner which implies that the scanning process is limited by the bandwidth of the compromised host, not by the latency of connection requests, as is often the case [17]. The use of scanning causes highly anomalous behavior as it generates a lot of traffic that differs from normal traffic. This makes the worms easier to detect.

4.1.2.Target Lists

Target discovery can also be carried out through the use of target lists. Worms utilizing such lists are often referred to as hit list worms and are characterized by their extremely rapid spreading speed. One example is the use of pre-generated target lists where a set of hosts known or suspected to be vulnerable to attack is gathered in advance and is included in the actual worm payload. A small target list of this kind could be used to accelerate the spreading of a scanning worm, while a complete list could create a flash worm which is further elaborated in section 3.4.3. An externally generated target list is a target list not included in the worm's payload, but maintained by a separate server. The list can be downloaded to infected machines in order to select new victims. An externally generated target list located at a central server makes it easy to issue updated target lists, but at the same time, if the central server is compromised the worm may be prevented from further propagation [15]. Yet another example of a target list is the host-based lists in which the worm utilizes information stored on the infected host to decide which hosts to attack next. Worms utilizing host-based lists for target discovery are called topological worms.

4.1.3.Passive Monitoring

Worms using a passive monitoring technique are not actively searching for new victims. Instead, they are waiting for new targets to contact them or rely on the user to discover new targets. Although passive worms tend to have a slow propagation rate, they are often difficult to detect because they generate modest anomalous reconnaissance traffic.

4.2. Propagation Carrier and Distribution Mechanism

There are three possible methods by which a worm can propagate from an infected host to an uninfected one [15].

4.2.1. Self-Carried

A self-carried worm transmits itself as part of the infection process. This mechanism is commonly used when the initial attack is directly followed by the worm payload transmission, as is the case with self-activating and topological worms.

4.2.2. Second Channel

Some worms require a second communication channel in order to complete the infection process. One example is to have the victim host request the transfer of the actual worm code to complete the infection.

4.2.3. Embedded

An embedded worm transmits itself as part of a normal communication channel by appending itself to, or replacing, an existing payload. This yields modest anomalous traffic related to propagation and could be combined with a stealthy target discovery mechanism, like the passive monitoring mechanism described in the previous section, in order to create a stealthy worm.

4.3. Activation

The means by which a worm is activated on a newly infected host drastically affects its propagation speed.

4.3.1. Human Activity-Based Activation

Some worms are activated when the user performs some activity, like resetting the machine, logging onto the system and thereby running the login scripts or executing a remotely infected file. Evidently, such worms do not spread very rapidly.

4.3.2. Scheduled Process Activation

A faster spreading speed than the previous activation method is achieved by worms that rely on some scheduled process for activation. An example is automatic software updates, which can be used to install and run malicious software (e.g., a worm). Earlier versions of automatic update services were more susceptible to this kind of attack as they rarely employed any authentication.

4.3.3. Self Activation

The fastest spreading worms are the ones that are able to activate themselves by initiating their own execution as soon as the infection process is completed. This is done by exploiting vulnerabilities in a service that is always running and available, or in the libraries that these services use. The worms activate themselves by attaching themselves to the running service or by executing commands using the permissions associated with those services.

4.4. Payload

The worm code not related to propagation is called the worm payload. It can vary significantly depending on the goals of the worm's author. Some examples are presented in this section.

4.4.1. None/Nonfunctional

The most common payload is actually no or a nonfunctional payload. Even with no payload, the worm can still consume considerable network and computer resources, as well as advertising vulnerable hosts.

4.4.2.Remote Control

Some payloads can open backdoors on victim machines in order to make remote control of the captured machines possible by bypassing the usual security access procedures. By introducing a trojan horse to the infected machine, it is possible to gain access to files that normally require certain user privileges [17].

4.4.3. Denial of Service (DoS)

A commonly used payload is to issue a Denial of Service attack against one or several web sites. The effect of a DoS attack increases with the number of nodes participating in the attack. A large worm network can cause large damage by issuing a Distributed DoS (DDoS) attack, where all the worm nodes simultaneously launch attacks against the same web site.

4.4.4. Data Collection

An increasing amount of sensitive information is stored electronically these days. Worm payload can search for this type of information (e.g., credit card numbers). Findings could be encrypted and transmitted through various channels.

4.4.5. Data Damage

Data damage is likely to become a popular worm payload, like it has been for some time for computer viruses. It can be used to erase or manipulate data on the infected host, or even to encrypt data in order to extort the owner of the information.

II. 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.

5.1.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.

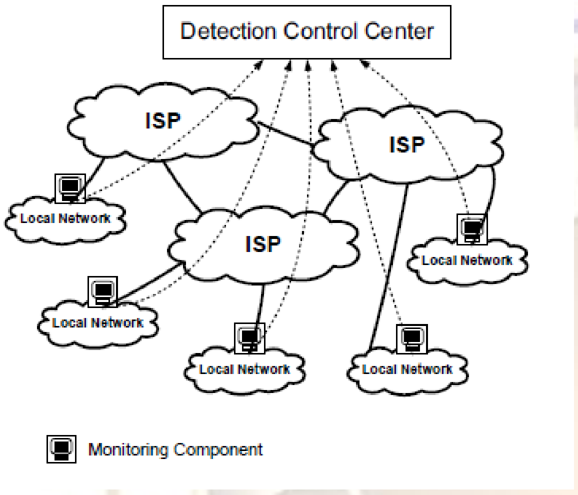

Fig.1.

5.2. 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, how-ever, prone to false alarms because the scan traffic can be caused by other reasons (such as DDOS and soft-ware 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.

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.

5.2.1.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.

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 *Ti* in Equation , we consider there is an anomaly.

$\Delta v_{i+1} - E[\Delta v_i] > T_i$ ---------------------1)

$E[\Delta v_i] = \frac{1}{k}\sum_{j=i-k}^{i-1} \Delta V_j$ ---------------------2)

$T_i = \gamma * \sqrt{\frac{1}{k}\sum_{j=i-k}^{i-1}(\Delta V_j - E[\Delta V_i])}$---------3)

where *Vi* is the number of victims detected by the system up to time tick *i*. ¢*Vi*+1 = *Vi*+1¡*Vi*, which denotes the number of new victims detected from time tick *i* to time tick *i*+1. *E*[¢*Vi*] 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. ° is a constant value called threshold ratio. *Ti* 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 *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 Figure. The monitoring components gather scan packets to the detection networks, and use SDR 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.

5.2.2.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 $\gamma$, learning time K and
   threshold ratio $\gamma$ .
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. alter a worm attack.
```

5.2.3.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.

The performance can be estimated by the following criteria.

1) Modeling the Number of victim
2) Requirements for Detection Network size
3) Traffic collection
4) Parameter selection

The following figures shows that detection time of different scan techniques using detection network.

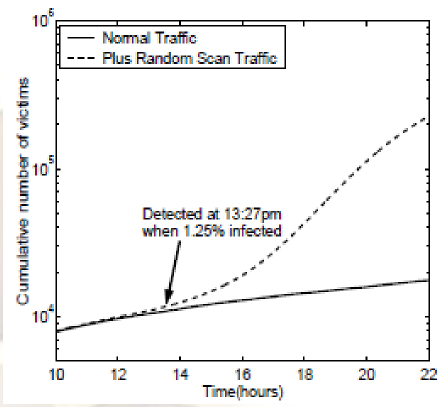

Fig. 2a) Detection of a Random scan

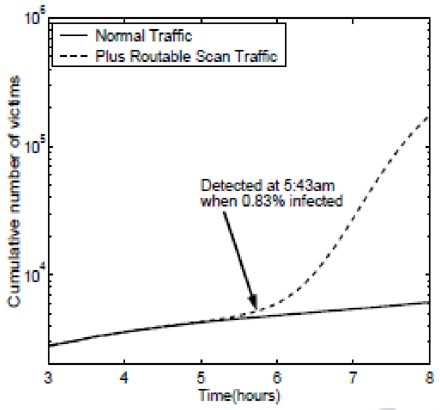

Fig. 2b) Detection of a Routable scan

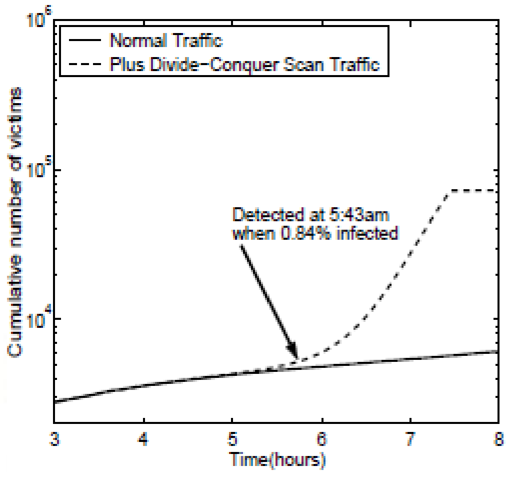

Fig. 2c) Detection of a Divide-Conquer scan

5.2.4.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. 2(a) shows the detection time for random scan worm. The worm startsat 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. 2(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.2(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. 3(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. he 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. 2(b) and Fig. 2(c) show similar plots for routable scan and divide conquer scan worms respectively. To understand how - (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. 3(a), we vary - from 1*:*3 £ 109 to 232 when *N* = 500*;* 000. The worm can be detected before 1.4% of vulnerable machines are infected in most cases. we vary *N* from 0*:*1£106 to 2*:*0£106 when - = 232. It shows that worms can be detected before 2% of vulnerable machines are infected.

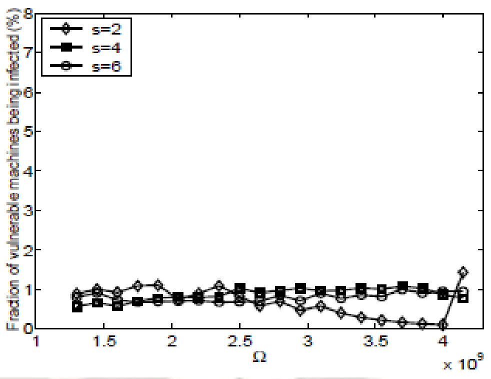

Fig. 3a) Fraction of Vulnerable machines being infected vs Ω

11

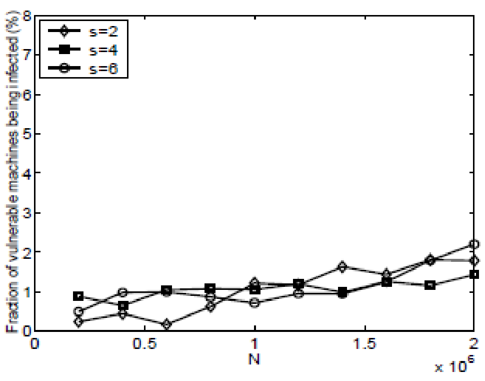

Fig. 3b) Fraction of Vulnerable machines being infected Vs N

III. CONCLUSION

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 clearly mentioned how the worms will be effected and the characteristics of different types of worms along with the architecture and along with the algorithm to identify the effected worms in the network dynamically by using different scan techniques like random scan, routable scan, divide-conquer scan. Further this paper, can be extend to detecting the worms in world wide web.

References

[1] D. Moore, C. Shannon, and J. Brown, “Code-Red: A Case Study on the Spread and Victims of an Internet Worm,” Proc. Second Internet Measurement Workshop (IMW), Nov. 2002.

[2] D. Moore, V. Paxson, and S. Savage, “Inside the Slammer Worm,” Proc. IEEE Magazine of Security and Privacy, July 2003.

[3] CERT, CERT/CC Advisories, http://www.cert.org/advisories/, 2010.

[4] P.R. Roberts, Zotob Arrest Breaks Credit Card Fraud Ring, http:// www.eweek.com/article2/0,1895,1854162,00.asp, 2010.

[5] D.Haand H.Ngo, “OntheTrade-Offbetween Speedand Resiliency of Flash Worms and Similar Malcodes,”Proc.Fifth ACM Workshop Recurring Malcode (WORM),Oct.2007.

[6] 6)Yang,S.Zhu, andG.Cao, “Improving Sensor Network Immunity under WormAttacks: A Software diversity Approach Proc.ACMMobiHoc,May2008.

[7] C. Zou, D.Towsley, and W.Gong ,“Email Worm Modeling and Defense ,”Proc.13th Int’l Conf. Computer Comm.and Networks (ICCCN),Oct.2004.

[8] W.Yu,S.Chellappan,C.Boyer,andD.Xuan,“Peer-to-Peer System-Based Active Worm Attacks: Modeling and Analysis,” Proc. IEEEInt’l Conf. Comm.(ICC),May2005.

[9] Z.S.Chen,L.X.Gao,andK. Kwiat,“Modeling the spread of Active Worms, ”Proc. IEEE INFOCOM ,Mar.2003.

[10] J.Wu,S.Vangala, and L.X.Gao, “An Effective Architecture and Algorithm for Detecting Worms with Various Scan Techniques,” Proc.11thIEEENetworkandDistributedSystemSecuritySymp. (NDSS),Feb.2004.

[11] S.Staniford, D.Moore, V.Paxson, and N.Weaver, “The Top Speed of Flash Worms,” Proc.Second ACMConf .Computer and Comm. Security(CCS) Workshop Rapid Malcode(WORM),Oct.2004.

[12] Y.Li,Z. Chen ,and C.Chen, “Understanding Divide-Conquer-Scanning Worms,” Proc.Int’ lPerformance Computing and Comm. Conf.(IPCCC), Dec.2008.

[13] Z.S.Chen, L.X.Gao, and K.Kwiat, “Modeling the Spread of Active Worms, ”Proc. IEEE INFOCOM,Mar.2003.

[14] Dynamic Graphs of the Nimda Worm, http://www.caida.org/dynamic/analysis/security/nimda, 2010.

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

[16] Nicholas Weaver, Vern Paxson, Stuart Staniford, and Robert Cunningham. A Taxonomy of Computer Worms. 2003. http://www.cs.unc.edu/~jeffay/courses/nidsS05/attacks/ paxson-worm-taxonomy03.pdf.

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

Ravinder Nellutla B.Sc from KaKatiya University Warangal, Master of Computer Applications from KaKatiya University Warangal, M.Tech Computer Science Engineering from Balaji institute of Engineering and Sciences, Narsampet, Warangal, Currently working as Asst.Prof. at Kamala institute of technology and science, Singapur, Huzurabad, Karimnagar. His interested subjects include Programming languages, network security and Data base Concepts.

.

Vishnu Prasad Goranthala M.Tech Computer Science and Engineering from JNTU,Hyderabad, Master of Computer Applications from Osmania University ,BSc from KaKathiya University Warangal, Currently he is working as an Associate Prof, at Balaji Institute of Engineering & Sciences, Narsampet, Warangal., and has 9+ years of experience in Academic. His research areas include Databases, Programming Languages and Mobile computing, Information Security, Cryptography, and Network Security.

Fasi Ahmed Parvez currently he is the head of Department of CSE & IT in Balaji Institute of Engineering & Sciences, Narsampet, Warangal., Parvez had several years of Experience in Academic. His research areas of interest include Data mining, Databases, Information security.