Analyzing The Behaviour And Propagation Traffic Generated By Active Worms

M. Shashidar[1], K. Indraneel[2], Nagaraju Mamillapally[3]
[1]*M.Tech(CSE), Sri Kottam Tulasi Reddy Memorial College of Engineering, Kondair, Andhra Pradesh, India*
[2]*Asso.Professor, Sri Kottam Tulasi Reddy Memorial College of Engineering, Kondair, Andhra Pradesh, India*
[3]*Asst.Professor, Adarsh PG College of Computer Sciences, Mahabubnagar, Andhra Pradesh, India*

Abstract

Because of the ability of self propagation, active worms cause major threats to the computers connected over the internet. In an automated fashion these worms continuously propagates over the internet causes the computers to compromise and pose major security threats. There is a necessity of identifying such worms at some stage, stop its propagation and destruction causing by them. This can be done by studying its behaviour and implementing certain detection schemes.

In this paper we analyze various computer worms with their behaviour and the propagation traffic generated by them.

1. Introduction

Computer worms are self-propagating malicious codes spread themselves without any human interaction and launch the most destructive attacks against computer networks like launching massive Distributed Denial-of-Service attacks that disrupt the Internet utilities, access confidential information that can be misused through large-scale traffic sniffing, key logging etc., They destroy data that has a high monetary value, and distribute large-scale unsolicited advertisement emails or software.

Due to the substantial damage caused by worms in the past years, there have been significant efforts on developing detection and defense mechanisms against worms. A network-based worm detection system plays a major role by monitoring, collecting, and analyzing the scan traffic (messages to identify vulnerable computers) generated during worm attacks. In this system, the detection is commonly based on the self-propagating behavior of worms that can be described as follows: After a worm-infected computer identifies and infects vulnerable computers on the Internet, this newly infected computer will automatically and continuously scan several IP addresses to identify and infect other vulnerable computers. As such, numerous existing detection schemes are based on a tacit assumption that each worm-infected computer keeps scanning the Internet and propagates itself at the highest possible speed. Furthermore, it has been shown that the worm scan traffic volume and the number of worm-infected computers exhibit exponentially increasing patterns.

After an introductory terminology is presented, worm characteristics during target finding and worm transferring phases are identified. Depending on where the detection is implemented, they may construct different views of worm propagation behaviours, so there may be differences in the scope of their defences.

Such a technology have been identified various phases like activation, false alarm, false positive, false negative, infection, target finding, threshold, transfer of worms life makes them to detect.

In this paper we study and analyze the behaviour of various worms like C-Worm, Morris Worm, Code-Red Worm and Slammer Worm and also identified their life cycle based on their propagation over the internet.

2. Background and Related Work

Computer Active worms are similar to biological viruses in terms of their infectious and self-propagating nature. They propagate into computers in the botnet which are identified as vulnerable, 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.

After many Internet-scale worm incidents in recent years, it is clear that a simple self-propagating worm can quickly spread across the Internet and cause severe damage to our society. Facing this great security threat, we need to build an early detection system that can detect the presence of a worm in the Internet as quickly as possible in order to give people accurate early warning information and possible reaction time for counteractions. based on the idea of "detecting the trend, not the burst" of monitored illegitimate traffic, we present a "trend detection" methodology to detect a worm at its early propagation stage by using Kalman filter estimation, which is robust to background noise in the monitored data. In

addition, for uniform-scan worms such as Code Red, we can effectively predict the overall vulnerable population size, and estimate accurately how many computers are really infected in the global Internet based on the biased monitored data. For monitoring a non uniform scan worm, especially a sequential-scan worm such as Blaster, we show that it is crucial for the address space covered by the worm monitoring system to be as distributed as possible.

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). In addition, worms use different scan strategies during different stages of propagation. In order to increase propagation efficiency, they use a local network or hit list to infect previously identified vulnerable computers at the initial stage of propagation. They may also use DNS, network topology, and routing information to identify active computers instead of randomly scanning IP addresses. They split the target IP address space during propagation in order to avoid duplicate scans and studied a divide-conquer scanning technique that could potentially spread faster and stealthier than a traditional random-scanning worm. Ha and Ngo formulated the problem of finding a fast and resilient propagation topology and propagation schedule for Flash worms. Yang et al. studied the worm propagation over the sensor networks.

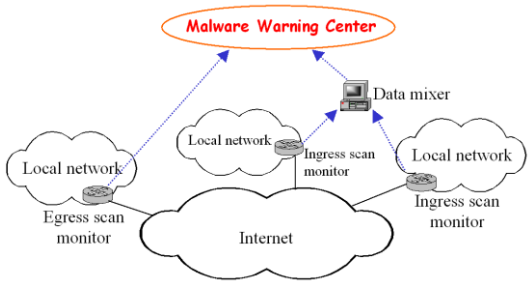

Figure 1: A Generic Worm Monitoring System

Worm detection has been intensively studied in the past and can be generally classified into two categories: "host based" detection and "network-based" detection. Host based detection systems detect worms by monitoring, collecting, and analyzing worm behaviours on end-hosts. Since worms are malicious programs that execute on these computers, analyzing the behaviour of worm executables plays an important role in host-based detection systems. In contrast, network-based detection systems detect worms primarily by monitoring, collecting, and analyzing the scan traffic (messages to identify vulnerable computers) generated by worm attacks.

In order to rapidly and accurately detect Internet-wide large-scale propagation of active worms, it is imperative to monitor and analyze the traffic in multiple locations over the Internet to detect suspicious traffic generated by worms. The widely adopted worm detection framework consists of multiple distributed monitors and a worm detection center that controls the former.

3. Analyze Worms Behaviour

In this section we look at one of the new class of active self propagation worm, the Camouflaging worm (C-Worm) and then discuss four more recent Internet worms: Morris, Code Red, and Slammer based on their characteristics.

3.1 Camouflaging Worm

Different from the above worms, which attempt to accelerate the propagation with new scan schemes, the C-Worm studied in this paper aims to elude the detection by the worm defense system during worm propagation. Closely related, but orthogonal to our work, are the evolved active worms that are polymorphic, in nature. Polymorphic worms are able to change their binary representation or signature as part of their propagation process. This can be achieved with self-encryption mechanisms or semantics-preserving code manipulation techniques.

The C-Worm also shares some similarity with stealthy port-scan attacks. Such attacks try to find out available services in a target system, while avoiding detection. It is accomplished by decreasing the port scan rate, hiding the origin of attackers, etc. Due to the nature of self propagation, the C-Worm must use more complex mechanisms to manipulate the scan traffic volume over time in order to avoid detection.

The C-Worm camouflages its propagation by controlling scan traffic volume during its propagation. The simplest way to manipulate scan traffic volume is to randomly change the number of worm instances conducting port-scans.

In order to effectively evade detection, the overall scan traffic for the C-Worm should be comparatively slow and variant enough to not show any notable increasing trends over time. On the other hand, a very slow propagation of the C-Worm is also not desirable, since it delays rapid infection damage to the Internet. Hence, the C-Worm needs to adjust its propagation so that it is neither too fast to be easily detected, nor too slow to delay rapid damage on the Internet. To regulate the C-Worm scan traffic volume, we introduce a control parameter called attack probability P(t) for each

worm-infected computer. P(t) is the probability that a C-Worm instance participates in the worm propagation (i.e., scans and infects other computers) at time t. Our C-Worm model with the control parameter P(t) is generic.

P(t) = 1 represents the cases for traditional worms, where all worm instances actively participate in the propagation. For the C-Worm, P(t) needs not be a constant value and can be set as a time-varying function. In order to achieve its camouflaging behaviour, the C-Worm needs to obtain an appropriate P(t) to manipulate its scan traffic. Specifically, the C-Worm will regulate its overall scan traffic volume such that:

- It is similar to non worm scan traffic in terms of the scan traffic volume over time,
- It does not exhibit any notable trends,
- The average value of the overall scan traffic volume is sufficient to make the C-Worm propagate fast enough to cause rapid damage.

The basic idea is to estimate the percentage of computers that have already been infected over the total number of IP addresses as well as M(t), through checking a scan attempt as a *new hit* (i.e., hitting an uninfected vulnerable computer) or a *duplicate hit* (i.e., hitting an already infected vulnerable computer). This method requires each worm instance (i.e., infected computer) to be marked indicating that this computer has been infected. Thus, when a worm instance (for example, computer *A*) scans one infected computer (for example, computer *B*), then computer *A* will detect such a mark, thereby becoming aware that computer *B* has been infected. Through validating such marks during the propagation, a C-Worm infected computer can estimate M(t). There are other approaches to achieve this goal, such as incorporating the Peer-to-Peer techniques to disseminate information through secured IRC channels.

3.2 Morris Worm

The Morris worm was one of the first Internet worms whose devastating effect gained the wide attention of the media. Morris worm was launched in November 1988 by Robert Tappan Morris, who was a student at Cornell University at the time. It is the first known worm to exploit the buffer overflow vulnerability. It targeted send mail and finger services on DEC VAX and Sun 3 hosts. Based on the creator's claim, the Morris worm was not intended to cause any harm, but was designed to discover the number of the hosts on the Internet. The worm was supposed to run a process on each infected host to respond to a query if the host was infected by the Morris worm or not. If the answer was yes, the infected host should have been skipped; otherwise, the worm would copy itself to the host. However, a flaw in the program caused the code to copy itself multiple times to already infected machines, each time running a new process, slowing down the infected hosts to the point that they became unusable.

The Morris worm was a mixture of sophistication and naivety. It had a simple overall design: look at a computer's system configuration to find potential neighbors, invade them, and try to minimize the number of invasions on any machine. The worm used heuristic knowledge about Internet topology and trust relationships to aid its spread, and it targeted two different machine architectures. Its cleverness in finding potential attack targets made it especially effective, but it also took on the time consuming task of guessing passwords on individual user accounts, which gave it an "attack in depth" aspect. Nonetheless, it became a victim of its own success as it was unable to control its exponential growth. With no global information and no point of control, the Morris worm ran rampant.

- It attacked one operating system, but two different computer architectures with three distinct propagation vectors.
- It had several mechanisms for finding both potential nodes to infect, and information found in user accounts.
- It traversed trusted accounts using password guessing.
- It installed its software via a two-step "hook and haul" method that required the use of a C compiler, link loader, and a call back network connection to the infecting system.
- It attempted to limit the reinfection rate on each node (but not the total number).
- It attempted to run forever on as many nodes as possible.

3.3 Code - Red Worm

Then on July 12, 2001, the Code-Red I worm began to exploit the aforementioned buffer-overflow vulnerability in Microsoft's IIS web servers. Upon infecting a machine, the worm checks to see if the date (as kept by the system clock) is between the first and the nineteenth of the month. If so, the worm generates a random list of IP addresses and probes each machine on the list in an attempt to infect as many computers as possible. However, this first version of the worm uses a static seed in its random number generator and thus generates identical lists of IP addresses on each infected machine.

This version spread slowly, because each infected machine began to spread the worm by probing machines that were either already infected or impregnable. On the 20th of every month, the worm is programmed to stop infecting other machines and proceed to its next attack phase in which it launches a Denial-of-Service attack

against www.whitehouse.gov. The worm is dormant on days of the month following the 28th.

On July 13th, Ryan Permeh and Marc Maiffret at eEye Digital Security received logs of attacks by the worm and worked through the night to disassemble and analyze the worm. They christened the worm "Code-Red" both because the highly caffeinated "Code Red" Mountain Dew beverage fueled their efforts to understand the workings of the worm and because the worm defaces some web pages with the phrase "Hacked by Chinese". There is no evidence either supporting or refuting the involvement of Chinese hackers with the Code-Red I worm. The first version of the Code-Red worm (Code-Red I v1) caused little damage. Although the worm's attempts to spread itself consumed resources on infected machines and local area networks, it had little impact on global resources.

The Code-Red I v1 worm is memory resident, so an infected machine can be disinfected by simply rebooting it. However, the machine is still vulnerable to repeat infection. Any machines infected by Code-Red I v1 and subsequently rebooted were likely to be reinfected, because each newly infected machine probes the same list of IP addresses in the same order. At approximately 10:00 UTC in the morning of July 19th, 2001, we observed a change in the behavior of the worm as infected computers began to probe new hosts. At this point, a random-seed variant of the Code-Red I v1 worm began to infect hosts running unpatched versions of Microsoft's IIS web server. The worm still spreads by probing random IP addresses and infecting all hosts vulnerable to the IIS exploit. Unlike Code-Red I v1, Code- Red I v2 uses a random seed in its pseudo-random number generator, so each infected computer tries to infect a different list of randomly generated IP addresses at an observed rate of roughly 11 probes per second (pps). This seemingly minor change had a major impact: more than 359,000 machines were infected with Code-Red I v2 in just fourteen hours. Because Code-Red I v2 is identical to Code-Red v1 in all respects except the seed for its pseudo-random number generator, the only direct damage to the infected host is the "Hacked by Chinese" message added to top level web pages on some hosts. However, Code-Red I v2 had a greater impact on global infrastructure due to the sheer volume of hosts infected and probes sent to infect new hosts. Code-Red I v2 also wreaked havoc on some additional devices with web interfaces. Although these devices were not susceptible to infection by the worm, they either crashed or rebooted when an infected machine attempted to send them the unusual http request.

Like Code-Red I v1, Code-Red I v2 can be removed from a computer simply by rebooting it. However, rebooting the machine does not prevent reinfection once the machine is online again. On July 19th, the number of machines attempting to infect new hosts was so high that many machines were infected while the patch for the vulnerability was being applied. On August 4, 2001, an entirely new worm, CodeRed II began to exploit the buffer-overflow vulnerability in Microsoft's IIS web servers. Although the new worm is completely unrelated to the original Code-Red I worm, the source code of the worm contained the string "CodeRed II" which became the name of the new worm. Ryan Permeh and Marc Maiffret analyzed CodeRed II to determine its attack mechanism. When a worm infects a new host, it first determines if the system has already been infected. If not, the worm initiates its propagation mechanism, sets up a "backdoor" into the infected machine, becomes dormant for a day, and then reboots the machine. Unlike Code-Red I, CodeRed II is not memory resident, so rebooting an infected machine does not eliminate CodeRed II. After rebooting the machine, the CodeRed II worm begins to spread. If the host infected with CodeRed II has Chinese (Taiwanese) or Chinese (PRC) as the system language, it uses 600 threads to probe other machines. On all other machines it uses 300 threads. CodeRed II uses a more complex method of selecting hosts to probe than Code-Red I. CodeRed II generates a random IP address and then applies a mask to produce the IP address to probe. The length of the mask determines the similarity between the IP address of the infected machine and the probed machine.

The CodeRed II worm is much more dangerous than Code-Red I because CodeRed II installs a mechanism for remote, administrator-level access to the infected machine. Unlike Code-Red I, CodeRed II neither defaces web pages on infected machines nor launches a Denial-of-Service attack. However, the backdoor installed on the machine allows any code to be executed, so the machines could be used as "zombies" for future attacks (Denial-of-Service or otherwise).

3.4 Slammer Worm

Slammer (sometimes called Sapphire) was the fastest computer worm in history. As it began spreading throughout the Internet, the worm infected more than 90 percent of vulnerable hosts within 10 minutes, causing significant disruption to financial, transportation, and government institutions and precluding any human-based response. Slammer began to infect hosts slightly before 05:30 UTC on Saturday, 25 January 2003, by exploiting a buffer-overflow vulnerability in computers on the Internet running Microsoft's SQL Server or Microsoft SQL Server Desktop Engine (MSDE) 2000. David Litchfield of Next Generation Security Software discovered this underlying indexing service weakness in July 2002; Microsoft released a patch for the vulnerability

before the vulnerability was publicly disclosed (www.microsoft.com/security/slammer.asp). Exploiting this vulnerability, the worm infected at least 75,000 hosts, perhaps considerably more, and caused network outages and unforeseen consequences such as canceled airline flights, interference with elections, and ATM failures.

Slammer's most novel feature is its propagation speed. In approximately three minutes, the worm achieved its full scanning rate (more than 55 million scans per second), after which the growth rate slowed because significant portions of the network had insufficient bandwidth to accommodate more growth. By comparison, Slammer was two orders of magnitude faster than the Code Red worm, which infected more than 359,000 hosts on 19 July 2001,2 and had a leisurely 37 minutes of population doubling time. While Slammer had no malicious payload, it caused considerable harm by overloading networks and disabling database servers. Many sites lost connectivity as local copies of the worm saturated their access bandwidths. If the worm had carried a malicious payload, attacked a more widespread vulnerability, or targeted a more popular service, its effects would likely have been far more severe.

4. Analyze the Propagation Traffic

In this section we study and analyze the propagation traffic of selected active worms.

4.1 Camouflaging Worm

To analyze the C-Worm, we adopt the epidemic dynamic model for disease propagation. Based on existing results, this model matches the dynamics of real-worm propagation over the Internet quite well. For this reason, similar to other publications, we adopt this model in our paper as well. Since our investigated C-Worm is a novel attack, we modified the original epidemic dynamic formula to model the propagation of the C-Worm by introducing the P(t)—the attack probability that a worm-infected computer participates in worm propagation at time t. We note that there is a wide scope to notably improve our modified model in the future to reflect several characteristics that are relevant in real-world practice.

Particularly, the epidemic dynamic model assumes that any given computer is in one of the following states: immune, vulnerable, or infected. An immune computer is one that cannot be infected by a worm; a vulnerable computer is one that has the potential of being infected by a worm; an infected computer is one that has been infected by a worm. The simple epidemic model for a finite population of traditional PRS worms can be expressed as

$$\frac{dM(t)}{dt} = \beta.M(t).[N - M(t)]$$

where M(t) is the number of infected computers at time t; N(=T. $P_1.P_2$) is the number of vulnerable computers on the Internet; T is the total number of IP addresses on the Internet; P_1 is the ratio of the total number of computers on the Internet over T; P_2 is the ratio of total number of vulnerable computers on the Internet over the total number of computers on the Internet; β =S/V is called the pair wise infection rate; and S is the scan rate defined as the number of scans that an infected computer can launch in a given time interval. We assume that at t=0, there are M(0) computers being initially infected and N-M(0) computers being susceptible to further worm infection. The C-Worm has a different propagation model compared to traditional PRS worms because of its P(t) parameter. Consequently, (1) needs to be rewritten as

$$\frac{dM(t)}{dt} = \beta.M(t).P(t).[N - M(t)]$$

Recall that P(t) = $\tilde{M}_C$/M(t) , _M (t) is the estimation of M(t) at time t, and assuming that M(t) = (1+∈).M(t), where ∈ is the estimation error, the (2) can be rewritten as

$$\frac{dM(t)}{d(t)} = \frac{\beta.\tilde{M}c.[N - M(t)]}{1+\in(t)}$$

We can derive the propagation model for the C-Worm as M(t)= $N\text{-}e^{-\frac{\beta.\tilde{M}c}{1+\in(t).t}}$(N-M(0), where M(0) is the number of infected computers at time 0. Assume that the worm detection system can monitor P_m(P_m∈[0,1]) of the whole Internet IP address space. Without loss of generality, the probability that at least one scan from a worm-infected computer (it generates S scans in unit time on average) will be observed by the detection system is $1\text{-}(1\text{-}P_m)^{P(t).S}$. We define that $M_A(t)$ is the number of worm instances that have been observed by the worm detection system at time t, then there are M(t)-M_A(t) unobserved infected instances at time t. At the worm propagation early stage, M(t)-M_A(t) ≅ M(t). The expected number of newly observed infected instances at t+δ (where δ is the interval of monitoring) is (M(t)-M_A(t)).[1- $(1\text{-}P_m)^{P(t).S}$] ≅ M(i)[1-$(1\text{-}Pm)^{P(t).S}$]. Thus, we have $M_A(t+\delta) = M_A(t)+M(t)[1-(1-P_m)^{P(t).S}$ Using simple mathematical manipulations, the number of worm instances observed by the worm detection system at time t is

$$\mathbf{M_A(t) = P(t).M(t).P_m = P_m.Mc/1+e(t)}$$

4.2 Morris Worm

Once launched, the worm moved from node to node using only itself and the infected node's local information. The worm did not receive information from other worms. This simplicity was probably a blessing and a curse, because it minimized the prerequisites for gaining a foothold, but it also made the worm difficult to control. Worms like the Morris follow these steps to establish a graph link.

a. Choose an endpoint.
b. Choose a vulnerable application.
c. Compute authentication information (if necessary).
d. Establish a network connection.
e. Control the new, remote worm instance using the remote application vulnerability to propagate the worm software by sending "hook" software source code and completion instructions, wait for the "haul" network connection from the endpoint, send worm body as binary load modules, and wait for the remote system to construct the worm and call back.
f. Go on to the next endpoint.

The worm needed software vulnerabilities to infect a new node, and it also needed to find likely targets for infection. The worm had the interesting property of being able to use two different vulnerabilities, one hidden in an email application. It also had a method for breaking into user accounts and spreading to sites trusted by those users.

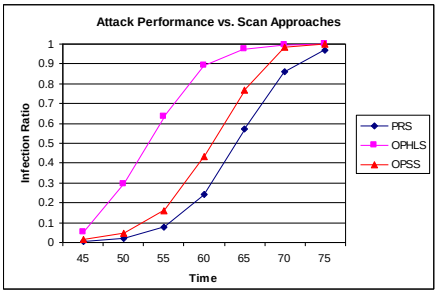

Figure 2: Propagation of Morris Worm

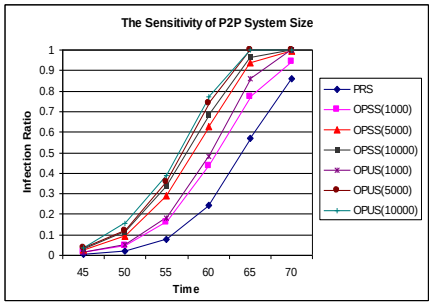

Figure 3: Sensitivity of attack to P2P System Size.

4.3 Code - Red Worm

Analysis of the Code-Red I worm covers the spread of the worm between July 4, 2001 and August 25, 2001. Before Code-Red I began to spread, we were collecting data in the form of a packet header trace of hosts sending unsolicited TCP SYN packets into our /8 network. When the worm began to spread extensively on the morning of July 19, we noticed the sudden influx of probes into our network and began our monitoring efforts in earnest.

Early on July 20, the filter was removed and we resumed packet header data collection. Although we collected data through October, we include data through August 25, 2001 in this study. No significant changes were observed in Code-Red I or CodeRed II activity between August 2001 and the pre-programmed shutdown of CodeRed II on October 1, 2001.

4.4 Slammer Worm

The worm's spreading strategy uses random scanning—it randomly selects IP addresses, eventually finding and infecting all susceptible hosts. Random-scanning worms initially spread exponentially, but their rapid new-host infection slows as the worms continually retry infected or immune addresses.

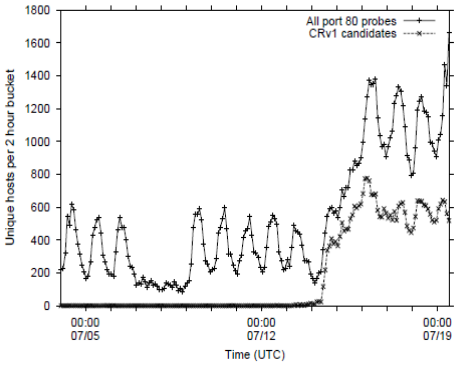

Figure 4: Background level of unsolicited SYN probes, beginning of Code-Red I worm spread.

Thus, as with the Code Red worm shown in Figure 6, Slammer's infected-host proportion follows a classic logistic form of initial exponential growth in a finite system.1,2 We label this growth behavior a *random constant spread* (RCS) model. Slammer's spread initially conformed to the RCS model, but in the later stages it began to saturate networks with its scans, and bandwidth consumption and network outages caused site-specific variations in its observed spread. Figure 10 shows a data set from the Distributed Intrusion Detection System project (Dshield; www.dshield.org) compared to an RCS model. The model fits extremely well up to a point where the probe rate abruptly levels out. Bandwidth

saturation and network failure (some networks shut down under the extreme load) produced this change in the probe's growth rate.

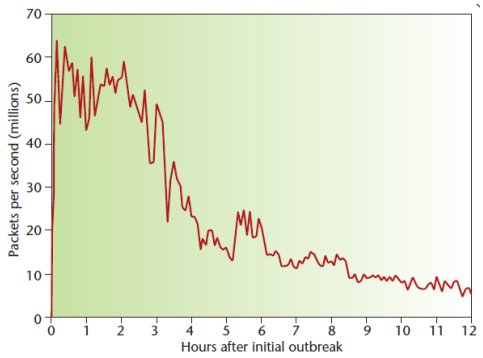

Figure 5: **Slammer propagation during the 12 hours after its release.**

5. Conclusion

Worms pose a serious and increasing threat to Internet security and stability. This being the case, system administrators must study the worm phenomenon and devise methods by which the spread of worms can be stopped. By examining historical and contemporary worms it becomes clear that, from a security standpoint, they merely represent variations on a few central themes.

Highlighting these commonalities allows for the synthesis of a platform-agnostic model of worm propagation that can then be systematically analyzed to determine where security technologies can be deployed to prevent such propagation from occurring. This analysis has shown that all is not lost in the fight against worms and that there are many commonly available security technologies that can be deployed to help prevent the spread of worms.

As demonstrated above, worm propagation is a multi-stage process in which a number of security technologies can be successfully deployed at each stage to help prevent, slow, or contain the spread of worms. This leads to the conclusion that mass worm outbreaks must be the result of generally lax security policies on a global scale rather than to a deficit in security technology.

6. References

[1]. Nagaraju Mamillapally, Venkatesh Gadege – " A Behavioural Study of Various Worms and their Detection Schemes", *Journal of Engineering, Computers & Applied Sciences (JEC&AS) ISSN No: 2319-5606 Volume 1, No.3, December 2012.*

[2]. Wei Yu, Xun Wang, Prasad Calyam, Dong Xuan, and Wei Zhao – "Modeling and Detection of Camouflaging Worm", IEEE Transactions on Dependable and Secure Computing, Vol. 8, No. 3, May/June 2011.

[3]. Pele Li, Mehdi Salour, And Xiao Su, San Jose State Uuniversity, "A Survey of Internet Worm Detection and Containment 1ST QUARTER 2008, Vol 10, No. 1.

[4]. 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.

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

[6]. E. Spafford, "The Internet Worm Program: An Analysis," *Comp.Commun. Rev.*, 1989.

[7]. "Morris (Computer Worm)," retrieved July 2007,http://en.wikipedia.org/wiki/Morris_worm M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[8]. G. P. Schaffer, "Worms and Viruses and Botnets, Oh My! Rational Responses to Emerging Internet Threats," *IEEE Sec. & Privacy*, vol. 4, 2006, pp. 52–58.