The definitive version is available at: http://dx.doi.org/10.1109/ICCCNT.2014.6963124

A Pseudo-Worm Daemon (PWD) for Empirical Analysis of Zero-Day Network Worms and Countermeasure Testing

Khurram Shahzad and Steve Woodhead
Internet Security Research Laboratory
Department of Electrical, Electronic and Computer Engineering
University of Greenwich
London, UK
{sk81,ws01}@gre.ac.uk

***Abstract*— The cyber epidemiological analysis of computer worms has emerged a key area of research in the field of cyber security. In order to understand the epidemiology of computer worms; a network daemon is required to empirically observe their infection and propagation behavior. The same facility can also be employed in testing candidate worm countermeasures. In this paper, we present the architecture and design of Pseudo-Worm Daemon; termed (PWD), which is designed to perform true random scanning and hit-list worm like functionality. The PWD is implemented as a proof-of-concept in C programming language. The PWD is platform independent and can be deployed on any host in an enterprise network. The novelty of this worm daemon includes; its UDP based propagation, a user-configurable random scanning pool, ability to contain a user defined hit-list, authentication before infecting susceptible hosts and efficient logging of time of infection. Furthermore, this paper presents experimentation and analysis of a Pseudo-Witty worm by employing the PWD with real Witty worm outbreak attributes. The results obtained by Pseudo-Witty worm outbreak are quite comparable to real Witty worm outbreak; which are further quantified by using the Susceptible Infected (SI) model.**

Keywords—worm, witty, scanning, hit-list, cyber

I. INTRODUCTION

Since the SQL Slammer Internet wide outbreak in 2003 [1], computer network worms have emerged as key security threat to the internet and national infrastructure. The key characteristics possessed by worms are; a rapid rate of propagation, ability to self–replicate and increased sophistication of worm's code, which have made them highly infectious and capable of causing a denial of service attack on the internet as in the case of SQL Slammer outbreak. The SQL Slammer worm is considered to be the fastest random-scanning worm in the history as it has achieved its full aggregate scanning rate, of over 55 million scans per seconds, only after 3 minutes of its release [1], while infecting 90% of susceptible machines within 10 minutes [1,2]. Although, it did not contain any malicious payload, the traffic it has generated created halted parts of internet by causing a denial-of- service attack.

Hit-list, Flash and Warhol worms are evolving categories of worms, capable of spreading even at faster rate than random scanning worm. Staniford et al. [3] simulated Warhol and Flash worm in 2002, and predicted that a Warhol worm, having an initial hit-list size of 10,000 and scanning rate of 100 scan/second in a population of 300,000 susceptible hosts in address space of 2^{32}, is capable of infecting all the hosts in 15 minutes. While a UDP based Flash worm with initial global size hit-list could saturate 95% of one million vulnerable hosts on the internet in 510 milliseconds [4]. Whilst other reported works [1,5] have presented analysis of worms based of reported worm outbreak data, and some reported work [4,6] have presented simulated worms, but no previously reported work has produced the empirical analysis of scanning, hit-list or flash worms on large scale network with real worm outbreak conditions. Hence there is a need to design, implement a pseudo-worm daemon with scanning and hit-list functionality, and to empirically analyze its spread on a

large scale network in order to find epidemiological trends of worm outbreaks.

With the aim of empirically analyzing the propagation of network worms, and testing potential countermeasures, this paper presents the architecture and design of PWD; capable of performing true random scanning and hit-list functionality. The key attributes of PWD are; its UDP based propagation, user- configurable random scanning pool, ability to contain user defined hit-list, authentication before infecting vulnerable machines and efficient logging of time of infection. This paper also presents the empirical analysis of Pseudo-Witty worm with real exploitable conditions on the Virtualized Malware Testbed (VMT) by employing PWD; while obtained results are quantified by using Susceptible Infected (SI) model [7,8].

The remainder of paper is presented as follows: Section II discusses the lexicon related to computer worms; Section III summarizes the relevant previous work related to empirical analysis of computer worms and modelling; Section IV details the basic design methodology, system design and implementation, and key characteristics of PWD; Section V presents the experimental methodology, network setup, and results and analysis of Pseudo-Witty worm outbreak; and finally Section VI concludes the paper with a discussion summarizing the findings and identifying any limitations, as well as summarizing potential future work in this area.

II. LEXICON

A lexicon has been presented for the clarification of the following terms, owing to their specific use in this paper.

A. Computer Worm

A computer worm is a program that self-replicates and self-propagates across a network, exploiting security or policy flaws in widely-used network services, without any human intervention [9]. For example, SQL Slammer, Witty, Code Red [10] etc. Table I shows different types of worms.

TABLE I. TYES OF COMPUTER WORMS

Worm	***Description***
Scanning	A scanning worm employs different scanning strategies (random [1], sequential [9], permutation [3] or importance [11]) to spread. Scanning refers to the process of probing a set of IP addresses to identify vulnerable hosts [9,12]. For example, Slammer, Code Red etc.
Hit-List	A worm that employs a pre-generated list of susceptible IP addresses to infect can be classified as hit-list worm, such as Witty [5].
Topological	A topological worm [9,12] uses an internally generated targets list of vulnerable hosts, which is created by finding local information on networks such as the /etc/hosts file on UNIX systems, or local topological information by using ARP caches tables and netstat.
Meta-Server/ Search	A meta-server worm [9,12] uses externally generated targets list of vulnerable hosts, which is maintained by a separate server, such as a matchmaking service's meta-server e-g. Gamespy or web searches using Google in order to find vulnerable targets.
BGP Routing	A BGP routing worm [6] uses BGP scanning techniques, which employ BGP routing tables to narrow the scanning addresses space. This type of worm is capable of targeting particular systems within specific geographic location such as a specific country, ISP or autonomous system, and can spreads 2 to 3 times faster than tradition random scanning worms.
Warhol	Warhol worm [3] is a hypothetical, simulated, and very fast spreading worm that uses combination of a hit list (which helps initial spread) and permutation scanning (which keeps its high infection rate higher than random scanning).
Flash	Staniford et al. [4] simulated the extension of Warhol worm, which they named, the Flash worm. The flash worm contains an initial global size hit list. They hypothesized that UDP based flash worm can infect 95 percent of one million vulnerable hosts in 510 ms, while a TCP based flash worm can infect the same population in 1.3s.

B. Zero-Day Worm

A zero-day worm is a type of worm that exploits a zero-day vulnerability that has not been patched or widely acknowledged at the point of exploitation [9,12]. For example, Code Red and Slammer both used exploit zero-day vulnerabilities.

C. Empirical Analysis

In this paper, empirical analysis is based on launching a pseudo-worm with real worm attributes in a real, isolated network and analyzing its behaviour.

D. Epidemiological Analysis

In this paper, epidemiological analysis is based on rate at which total number of worm susceptible hosts becomes infected with worm at any given time of worm outbreak.

III. RELATED WORK

A. Empicircal Analysis of Zero-Day Worms

Moore et al. [1] provided the analysis of Slammer worm based on reported worm outbreak data, while Shannon et al. [5] presented the analysis of Witty worm based on reported worm outbreak data. But, as far as the authors of this paper are aware, no previous published work except [2] (which reports Slammer worm outbreak), has presented an infection and propagation analysis of any fast random scanning worm, Hit-list or Flash worm in a real isolated network with real worm outbreak conditions; which forms basis of this research work.

B. Epidemiological Modeling of Computer Worms

Mathematical modeling helps us understand the epidemiology of worm outbreaks; which is further used to monitor and defend against spread of computer worms. Various authors have proposed such mathematical models [1, 3,15,16,17] based on models originally developed for biological epidemiological studies [7,14]. Susceptible-Infected (SI) [7,14] model is most widely reported biological

model, which models the epidemiology of infection by assuming population of hosts is of fixed size and relying on a deterministic contact coefficient to govern the differential between each step of the model. Of note is another work undertaken by Zesheng et al. [18] that reports the discrete time deterministic model of active worms (the AAWP model), which characterizes the propagation of worms that employ random scanning and local subnet scanning. It uses discrete time model and deterministic approximation to describe the spread of active worms.

The key advantage of mathematical models are their ability to model large network worm outbreaks with relatively low resource restrictions, but these mathematical models are limited in high level of network granularity due to their macroscopic nature.

IV. Pseudo-Worm System Design and Implementation

A. Basic Desing and Methodology

A random scanning worm such as Slammer, Code Red etc. uses pseudo random number generator to scan random IP addresses, whereas a hit-list worm such as Witty uses initially generated hit-list of susceptible hosts, embedded into it, to infect susceptible hosts on the internet. Upon initial infection, a UDP based random scanning worm generates and sends a number of connection requests (defined by an attacker in the worm algorithm) to a number of random IP addresses in a unit interval of time (seconds or minutes). Each new infected host, upon infection, follows the same process and starts scanning the further IP addresses, thereby, creating a chain reaction. Fig. 1 shows this worm infection process [15]. In each stage of infection, each infected host n further scans m no of hosts. In random scanning worm such as SQL Slammer, at first stage infection, one or two infected hosts starts the infection process; while in the case of hit-list scanning such as Witty, at first stage of infection, worm infected host contains lists of susceptible hosts, which will start the infection process at once.

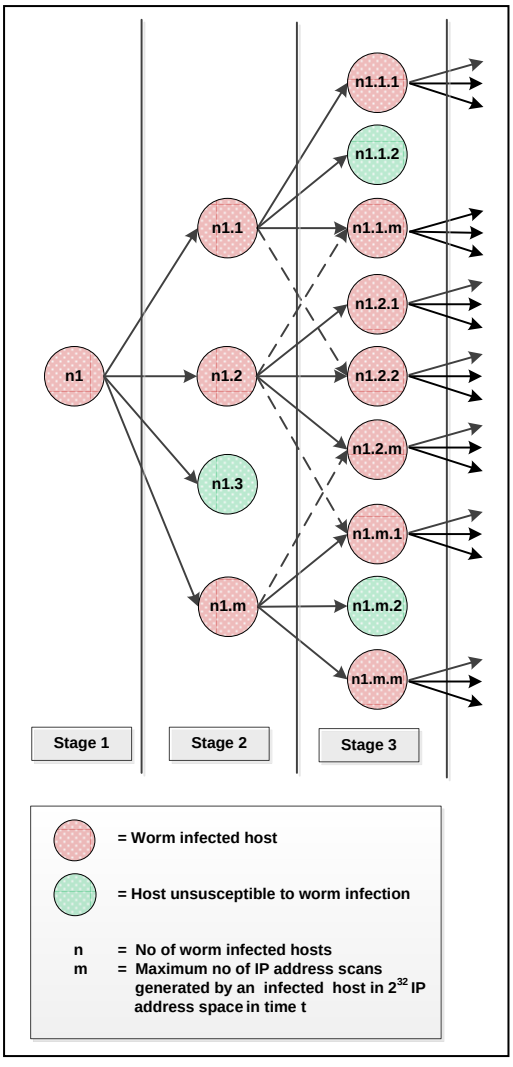

Fig. 1. Worm Infection Process

B. Pseudo-Worm System Design and Implementation

The pseudo-worm daemon is implemented in C programming language. C programming language is chosen to implement pseudo worm daemon due to its capability to access the systems low level functions, easily available open source, ease of use and platform independence. The basic design of pseudo-worm daemon consists of three key components:

1) UDP Client

A UDP client program is used to launch the worm. It can be installed on any host. It sends a UDP based connection request to UDP server by using a by using a single packet; which includes destination IP address, port no and authentication string.

2) UDP Server

A UDP server program is single threaded application that performs pseudo-worm like functionality. It can be installed on host in a network. Upon receiving connection request from UDP client on user defined port number and IP addresses, it authenticate the UDP client request to accept connection, sends the local time of connection to logging server and; turns it behaviour into client by sending further connection request to different destination IP addresses (generated by using random no generator). The rate of UDP datagrams generated per second and the IP address pool from which random destination IP addresses are chosen (either by random scanning or hit-list scanning from local file) are user-configurable parameters.

3) Logging Server

The logging server program is installed on any host in a network to collect time of infection from UDP servers. All hosts running UDP serves contains IP address of central logging server and upon infection, sends time of infection to logging server.

C. Characteristics of Pseudo-Worm Daemon

Following are key characteristics of the worm scanning daemon:

1) UDP based Propagation

A UDP based worm can propagate much faster than TCP based worm. The designed pseudo worm daemon uses UDP as its propagation mechanism, thereby making it similar in functionality to SQL slammer and Witty worm.

2) Pseudo Random Number Scanning

A pseudo random number generator (PRNG) is an algorithm for generating a sequence of numbers that approximates the properties of random numbers [19]. A random seed is used to initialize the PRNG. Various type of PRNG exists but the implementation of pseudo worm daemon presented in this paper, uses complementary multiply-with-carry (CMWC) [20] type of pseudo random number. CMWC method generates sequences of random integers based on an initial set from two to many thousands of randomly chosen seed values. The key advantages of the MWC method are: (a) it invokes simple computer integer arithmetic, (b) leads to very fast generation of sequences of random numbers with immense periods, ranging from around 2^{60} to $2^{2000000}$.

3) Hit-List Scanning

A pre-generated list of susceptible IP address can be included into text file. The PWD reads the text file and sends connection requests to those IP addresses before switching to random scanning mode, thereby imitating the functionality of a hit-list worm like Witty.

4) Containment

PWD has user- configurable random scanning pool, which can be defined inside it code. For example, generating IP addresses in one class A network size, or generating IP addresses in six class C networks, or generating IP address over whole internet space. Hence, its random IP generation can be contained in any network size according to needs of the security experiment.

5) Propagation rate

The number of random IP addresses scanned per second is defined as propagation rate of worm. For example, slammer generates 4000 IP addresses per second. PWD can be configured to scan at different propagation rate e.g. 100 scans per seconds, 500 scans per seconds etc.

6) Authentication

Any connection request from PWD contains an authentication string. Upon receiving a connection request, a host looks for authentication string, and if it finds the authentication string, it starts scanning new hosts. This authentication mechanism is included into PWD for safety reasons.

7) Logging and Reporting

Logging is process of recording events taking place in the execution of a system in order to provide an audit trial that can be used to understand the activity of the system and to diagnose problems [21]. The PWD includes logging server, which can be installed on any host. Upon infection, a UDP server sends the IP address of infected host and time of infection (up to microseconds) to central logging server. The central logging server stores this information in a text file which can be processed to find time of infection of all infected hosts in the network.

V. EXPERIMENTATION

A. Witty Worm

Shannon et al. [5] reported some key characteristics of Witty outbreak of 2004 which can be described as follows:

- Susceptible population of Witty Worm was 12, 000 or between 2 and 3 hosts per million of entire IPV4 address space.
- Witty has variable datagram size, with an Ethernet frame size between 796 and 1307 bytes.
- The average scanning rate of Witty was 357 datagrams per host per second during its entire infection period while the maximum recorded scanning rate of Witty was 970 datagrams per host per second.
- Witty also contains an initial hit-list of 110 hosts which were reported to be infected in first 10 seconds of infection out of which 38 hosts are transferring 9700 datagrams per host per second continuously for a period of an hour.

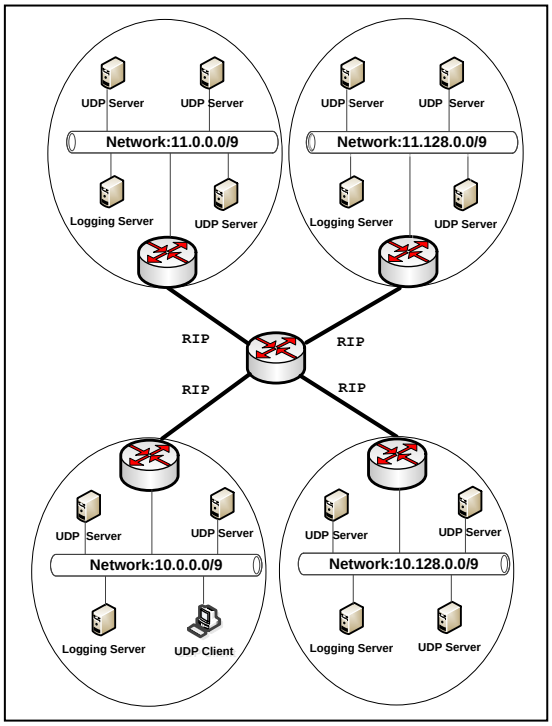

Fig. 2. Experimental Setup and Depolyment Architecture of PWD

B. Experimental Setup

In order to empirically analyze the behavior of Witty worm, an experimental test network was configured on Virtualized Malware Testbed (VMT) [2], comprising of a two Class A IP address space 10.0.0.0/8 and 11.0.0.0/8 but divided into four subnets; 10.0.0.0/10, 10.128.0.0/9, 11.0.0.0/9 and 10.128.0.0/9 as shown in Figure 2. These four subnets are connected through a central router by using RIP,

configured on Quagga. Four further Quagga based routers are implemented (one for each subnet). One Linux based virtual machine is running in each subnet to provide a DHCP service and logging service for pseudo-worm daemon. DSL is installed with the pseudo-worm daemon on each of the susceptible virtualized hosts. All hosts in the network are time synchronized by using the Network Time Protocol (NTP). The following fig. 2 shows experimental setup and; design and deployment architecture of PWD.

C. Experimental Methodology

As reported previously in section V.A, Witty has 3 hosts per million of the entire IPv4 addresses space were susceptible to infection with an average scan rate of 357 datagrams per infected host per second. A single class A network has 2^{24} hosts, and so 2 class A will contain $2^{24} * 2(3/1{,}000{,}000) = 101$ susceptible hosts. On this basis, 101 virtual machines with the PWD with Witty like attributes were deployed across the four subnets. Each worm daemon was configured to scan within two class A network (10.0.0.0/8, 11.0.0.0/8) at a scanning rate of 357 scans per host per second (average scan rate of Witty Worm as reported by Shannon et al.) while using an initial hit-list of one infected host along with first infected host.

D. Experimental Results

Fig. 3 shows the results of a set of three experiments conducted for pseudo witty worm outbreak. In the first experiment, all 101 susceptible machines got infected in 96.22 minutes. In the second experiment, all 101 susceptible machines got infected in 95.16 minutes. While in the third experiment, all 101 susceptible machines got infected in 94.46 minutes. It has been noted that in the middle of outbreaks of all three experiments, curves apart due to statistical variations and follows s-shaped random constant spread curve pattern [15].

Fig. 4 shows the comparison of real Witty worm outbreak of 2004 with pseudo-witty worm outbreak. The average of three pseudo-witty worm outbreaks is plotted against the real witty worm outbreak of 2004 as reported by Shannon et al. [5]. Shannon et al. reported that real witty worm infected 90% of its susceptible hosts within 45 minutes; while 100 % of infection took almost 140 minutes. But, the pseudo-witty experiments conducted by using PWD in test network, took almost 85 minutes to reach its 90 % of infection whereas 97 minutes on average to infection in all machines.

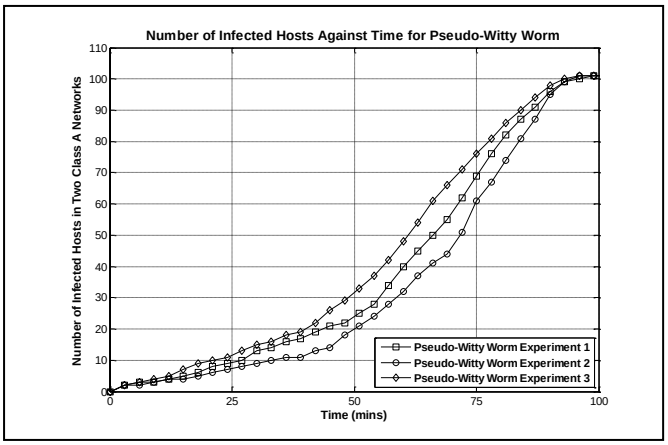

Fig. 3. Experimental Results of Psuedo-Witty Worm Outbreak

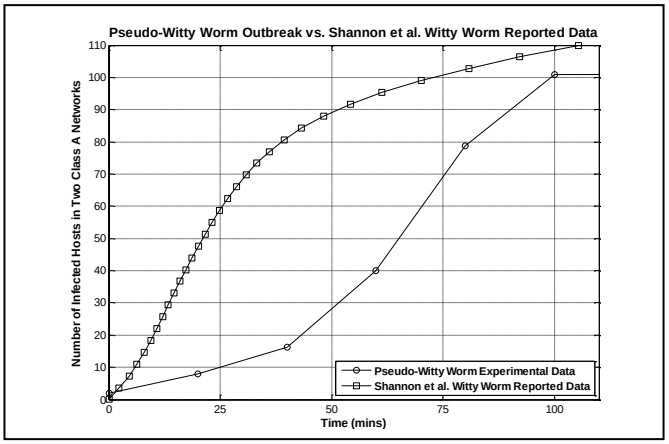

Fig. 4. Psuedo-Witty Worm Outbreak vs. Shanon et al. Witty Worm Reported Data

This difference is due to the fact that real Witty Worm outbreak contains an initial hit-list of 110 hosts, out of which 38 infected hosts are transferring 9700 datagrams per host per second continuously for a period of an hour; whereas Pseudo-Witty worm outbreak in the experiments used average scan rate of real Witty worm; which is 357 datagrams per host per second during its entire infection. But, still results of pseudo-witty worm outbreak are broadly comparable to real witty worm outbreak [5].

E. Epidemiological Analysis and Discussion

In order to further quantify the effectiveness of the countermeasures, we employed the susceptible-infected SI model [7], widely used to characterize the epidemiology of biological infections and proposed by Tidy et al [8] and Xiang et al [22] for similar use in characterizing the epidemiology of malware infections. The basis of this model is set out in Eqn 1, where I represents the number of infected hosts at time, t, N represents the total population size and β represents the contact coefficient. The particular solution to Eqn 1 is set out in Eqn 2, where I_0 represents the number of infected hosts at time t = 0.

Fig. 6 plots the best fit SI model curve against the experimental data set for experiment one (25% hosts vulnerable and no countermeasure), showing the Pearson's correlation coefficient [23], r, as well as the value of β for the SI model. Fig. 5 plots the SI model curve against the data from experiment three.

$$\frac{dI}{dt} = \beta . I . [N - I] \tag{1}$$

$$I(t) = \frac{I_0}{I_0 + [N - I_0] . e^{-N\beta t}} \tag{2}$$

It is worthy of note that the value of the correlation coefficient, r = 0.99186, is quite close to 1, indicating the

ability of the SI model to represent the experimental data for random scanning worms such as Witty. Furthermore, obtained experimental results also proved that Pseudo-Witty Worm outbreak follows random constant spread pattern.

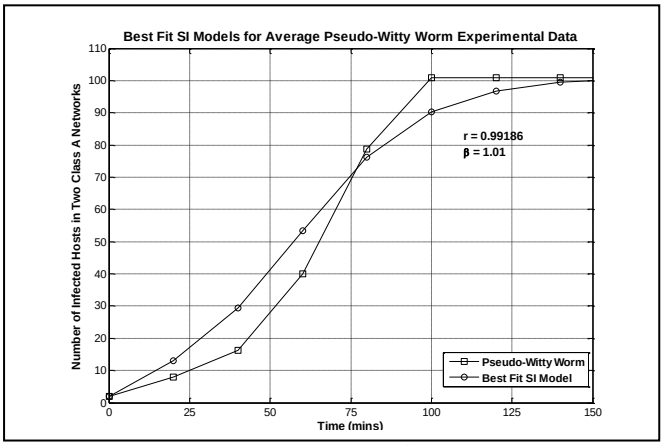

Fig. 5. Best Fit SI Model for Pseudo-Witty Worm Experimental Data

VI. Conclusion and Future Work

This paper presented an architecture and design of a pseudo-worm daemon (PWD) for cyber-epidemiological analysis of zero-day network worms. It has also reported its key characteristic of PWD such as; UDP based propagation, user- configurable random scanning pool, ability to contain user defined hit-list, authentication before infecting vulnerable machines and efficient logging of time of infection. We have conduced series of experiments by using PWD with Witty worm attributes. The experimental results are broadly comparable to real worm outbreak reported data. Furthermore, these results are quantified by using SI model. From base comparison of Witty worm results, it is concluded that PSD can be used as an effective tool to empirically analyze the propagation behavior of random scanning and hit-list worms and to test potential countermeasures.

The PWD uses only UDP as its propagation mechanism. In terms of future work, TCP based propagation can be integrated into its design in order to conduct epidemiological experiments of TCP based worms and to empirically analyze their propagation behavior. Furthermore, we also expect to experiment with a range of hypothetical worms such as Flash, Warhol [3,4], and to explore the applicability of PWD for characterizing their epidemiology.

Acknowledgment

Witty Worm Outbreak Dataset Provided by: The Cooperative Association for Internet Data Analysis (CAIDA) http://www.caida.org/data/passive/witty_worm_dataset.xml

References

[1] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, S and N. Weaver, “Inside the Slammer worm”, IEEE Security and Privacy 1, 4, July 2003.

[2] K. Shahzad, S. Woodhead, and P.Bakalis, “A virtualized network testbed for zero-day worm analysis”, in Proceedings of First International Conference of Advances in Security of Information and Communication Networks(SecNet2013), CICS, Springer, Cairo, Egypt, pp. 54–64, September 2013.

[3] S. Staniford, D. Moore, and V. Paxson, “How to own the internet in your free time”, in Proceedings of the 11th USENIX Security Symposium, USENIX, San francisco, CA, USA, pp. 149-167, 2002.

[4] S. Staniford, D. Moore, V. Paxson, and N.Weaver, “The top speed of flash worms”, in Proceedings of the 2nd ACM Workshop on Rapid Malcode (WORM), October 2004.

[5] S. Colleen, and D. Moore, “The spread of the witty worm”, IEEE Security and Privacy, pp. 46-50, 2004.

[6] Cliff C. Zou, D. Towsley, W. Gong, and S. Cai, “Routing worm: a fast, selective attack worm based on IP address information”, in Proceeedings of Workshop on Principles of Advanced and Distributed Simulation(PADS2005), ACM/IEEE/SCS, pp. 199-206, June 2005.

[7] W. Kermack and A. McKendrick, " A contribution to the mathematical theory of epidemics", in Proceedings of the Royal Society of London, vol. A, no 115, pp. 700-721,1927.

[8] Luc J Tidy, “ Investigation of zero-day worms with the aid of large-scale, high-fidelity simulation”, PhD Thesis, ISRL, University of Greenwich, May 2014.

[9] N. Weaver, V. Paxson, S. Staiford, and R. Cunningham, “A taxonomy of computer worms”, in Proceedings of the 2003 ACM workshop on rapid malcode WORM '03, New York, USA, ACM, pp. 11-18, 2003.

[10] D. Moore, C. Shannon, and J. Brown, “Code-Red: a case study on the spread and victims of an Internet worm”, In Proceedings of the Internet Measurement Workshop (IMW), Marseille, France, pp. 273–284, 2002.

[11] Z. Chen, and J. Chuanyi, “A self-learning worm using importance scanning”, *in* Proceedings of the ACM workshop on Rapid malcode WORM '05,. Alexandria, VA, USA, ACM, pp 22-29, 2005.

[12] P. Li, M. Salour, and Z. Su, “ A survery of Internet worm detection and containment”, IEEE Communication Surverys and Tutorials, 1st Quarter, 2008, Vol. 10, No. 1, pp. 20-35.

[13] L. Tidy, S.R. Woodhead, and J.C. Wetherall, “A large-scale zero-day worm simulator for cyber-epidemiological analysis”, International Journal of Advances in Computer Networks and Security, 3 (1), pp. 69-73, 2013.

[14] J. Frauenthal, "Mathematical Modeling in Epidemiology", Springer-Verlag, 1980.

[15] T. M. Chen and J.-M. Robert, “Worm epidemics in high-speed networks” in IEEE Computer, vol. 37(6), ppp 48–53, June 2004.

[16] M. Liljenstam, D.M. Nicol, V.H. Berk, and R.S. Gray, “Simulating realistic network worm traffic for worm warning system design and testing,”in Proceedings of ACM Workshop on Rapid Malcode 2003, pp. 24–33, 2003.

[17] C. C. Zou,W. Gong, D. Towsley, "Code Red worm propagation modeling and analysis" in Proceedings of 9th ACM Conference on Computer and Communication Security, pp. 138-147, Nov. 2002.

[18] Z. Chen, L. Gao, K. Kwiat, "Modeling the Spread of Active Worms",in Proceedings of IEEE INFOCOM 2003, pp. 1890 – 1900, 30 March-3 April 2003.

[19] E. Barker,W. Barker,W. Burr,W. Polk,and M. Smid, "Recommendation for Key Management", NIST Special Publication, NIST, July 2012, Retrieved on 27 April 2014.

[20] G. Marsaglia, A. Zaman, "A new class of random number generators", Annals of Applied Probability 1 (3), pp. 462–480, 1991.

[21] "Logfile", http://en.wikipedia.org/wiki/Logfile, Retrieved on 27 April 2014.

[22] Y. Xiang, X. Fan and W Zhu, “Propagation of active worms: a survey” in Computer Systems: Science and Engineering, volume 4, number 3, 2009.

[23] K. Pearson,“ Notes on regression and inheritance in the case of two parents”, in Proceedings of the Royal Society London , No 58, pp. 240-242, 1895.