Towards Automated Distributed Containment of Zero-day Network Worms

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

***Abstract*— Worms are a serious potential threat to computer network security. The high potential speed of propagation of worms and their ability to self-replicate make them highly infectious. Zero-day worms represent a particularly challenging class of such malware, with the cost of a single worm outbreak estimated to be as high as US$2.6Billion. In this paper, we present a distributed automated worm detection and containment scheme that is based on the correlation of Domain Name System (DNS) queries and the destination IP address of outgoing TCP SYN and UDP datagrams leaving the network boundary. The proposed countermeasure scheme also utilizes cooperation between different communicating scheme members using a custom protocol, which we term Friends. The absence of a DNS lookup action prior to an outgoing TCP SYN or UDP datagram to a new destination IP addresses is used as a behavioral signature for a rate limiting mechanism while the Friends protocol spreads reports of the event to potentially vulnerable uninfected peer networks within the scheme. To our knowledge, this is the first implementation of such a scheme. We conducted empirical experiments across six class C networks by using a Slammer-like pseudo-worm to evaluate the performance of the proposed scheme. The results show a significant reduction in the worm infection, when the countermeasure scheme is invoked.**

Keywords—malware, countermeasure, network worm, rate limiting

I. INTRODUCTION

Since the spread of the Morris worm in 1988 [1], computer network worms have become a persistent problem for internet users. Melissa, Code Red, Blaster, SQL Slammer, Conficker etc. did considerable damage to the internet community. SQL Slammer is considered to be the fastest random scanning worm in history as its infected population doubled in size every 8.5 seconds with 90 % of vulnerable machines infected within 10 minutes [2]. This worm achieved its full aggregate scanning rate, of over 55 million scans per second, only 3 minutes after it was released. It did not contain any malicious payload but the amount of traffic it generated, halted small parts of the internet for several hours. Flash, metamorphic and polymorphic worms are evolving categories of network worms considered a serious threat to the internet.

Staniford et al. [3] hypothesized in 2004 the top speed of a properly configured flash worm. They predicted that a UDP worm could saturate 95% of one million vulnerable hosts on the internet in 510 milliseconds. Today a worm can spread much more quickly than in 2004 due to the continuing increase in internet bandwidth. Due to the high speed and zero-day nature of many worms, traditional intrusion detection methods (i.e. generation and deployment of attack signatures) are ineffective [4]. These countermeasures also lack the ability to propagate malware warnings to uninfected sites in timely manner. Hence, in order to effect automatic detection and containment of zero-day worms, a rapid, accurate and distributed worm detection method is required.

In this paper, we report the design of a worm detection and containment scheme, which we then implemented as a proof-of-concept in C. The scheme can be deployed on the routers of enterprise networks. It uses the absence of a DNS lookup, prior to an outgoing TCP SYN or UDP datagram to a new destination IP address, as a behavioural signature to detect worm scanning activity. Upon detection of such behaviour, the scheme blocks further traffic from the originating host and sends an alert message using the Friends protocol to peer routers which belong to the scheme. To our knowledge, this is the first implementation of a hybrid worm detection and containment mechanism, based on a combination of resource limiting and leap ahead solutions.

The remainder of this paper is presented as follows: Section II discusses previously reported work on worm detection and prevention mechanisms and section III presents the basic methodology we employed for developing the reported countermeasure scheme. Section IV describes the high level system design. Section V describes the empirical evaluation of the prototype using the Slammer-like pseudo-worm. Section VI presents the experimental results while Section VII presents some analysis of the results. Section VIII concludes the paper and presents some possible future work directions.

II. RELATED WORK

Researchers have proposed various techniques for worm detection, mitigation and containment. For the purposes of this paper, we have chosen to adopt the classifications used by Porras et al [5]:

- Resourse limiting (RL) solutions;
- Leap ahead (LA) solutions;
- Predesigned-preventative (PP) solutions;

- Automatic signature generation (ASG) solutions;
- Mobile combat (MC) solutions.

A. Resource limiting solutions

RL solutions explore ways in which systems may delay worm propagation through the limiting of resources that fast worms are known to consume at high rates. Williamson [6] suggests that rate limiting the volume of outbound connections from a host to new machines can achieve a significant reduction in infection rate, without significantly hindering normal communications. Chen et al. [7] proposed another rate limiting scheme based on the assumption that a host infected by a scanning worm will generate a large number of failed TCP connection requests. Their scheme, which is implemented on the network router, attempts to rate limit outgoing datagrams from hosts that exhibits such behavior. Schechter et al [8], refined the outbound connection rate limiting concept. The scheme in [8] differs from that in [7] in two ways. First, it performs rate limiting exclusively on first contact outgoing connections for destination IP addresses that have not been visited previously. Second, it analyses both failed and successful TCP connection statistics. Gualtieri and Mosse [9] have proposed a scheme that dynamically calculates outbound connection rate limits on a per process basis. Ganger et al. [10] proposed that the analysis of network connections not facilitated through DNS lookups provide a relevant signature for identifying potential worm traffic. Wong et al. [11] explored the application of connection rate limiting to backbone routers by using the absence of DNS lookup statistics, suggesting that the throttling of IP-to-IP connection at the edge offers propagation reduction. The main advantages of RL solutions are their simplicity and ease of deployment but they are prone to a high rate of false positives as well as limited in terms of spreading malware threat warnings to uninfected hosts and networks.

B. Leap ahead solutions

LA solutions seek to spread malware warnings to network segments not yet affected. These strategies share cooperative information either hierarchically or using peer to peer based models. For example, Nojiri et. al. [12] proposed a cooperative alert sharing scheme using a "Friends protocol". Anagnostakis et. al. [13] proposed a variation of the LA scheme called Coverage, in which a node randomly selects a set of remote nodes to poll for worm reports at periodic intervals. The LA concept is effective in spreading malware warnings to uninfected network segments, but these solutions are limited in terms of their implementation in current networks.

C. Predesignated-preventative solutions

PP solutions are considered to be those approaches which are designed to disrupt the discovery of susceptible nodes within an address space, potentially by dynamically altering the connectivity of networks or end nodes in the presence of worm propagation. Briesemeister et al. [14] discussed the idea of percolation theory or epidemic spread in artificial scale-free networks to suggest how networks could be designed to delay the spread of propagating malware while still maintaining high reliability of network links. Gorman et al. [15] studied the use of scale-free properties within the autonomous system (AS) map of the internet, and proposed that the concentration of worm filtering services on the nodes with the highest connection density would yield the greatest benefit. Antonatos et.al. [16] proposed the concept of network address space randomization (NASR) against hit list worms which suggests frequently changing the IP addresses of hosts. Provos [17] suggested the placement of honeypot devices in a network that engage in slow connection dialogs as a method to dramatically slowing an aggressive worm's ability to discover susceptible hosts within an address space. Riordan et al. [18] proposed a honeypot worm detection system which they term Billy Goat which is widely deployed throughout IBM.

D. Automatic signature generation solutions

ASG solutions refer to approaches which filter incoming traffic to a network and generate signatures on detecting anomalous activity (such as a network worm). For example; Kim et al. [19] proposed a system that they named Autograph that automatically generates signatures for Bro or Snort for novel internet worms that propagate using TCP transport. Singh et al. [20] proposed an automated worm fingerprinting mechanism which they named Earlybird. Earlybird detects previously unknown worms and viruses based on two key behavioral characteristics: a common exploit sequence together with a range of unique sources generating infections and destinations being targeted. Although ASG solutions provide effective methods of worm detection, these solutions are prone to false positives and do not effectively spread malware warnings to uninfected networks.

E. Mobile combat solutions

MC solutions refer to approaches which involve an active strategy of interception and rapid patching. These techniques eliminate propagating malware by distributing a mobile self-replicating code module that searches out for signs of a malicious resident code and vaccinates infected machines. For example, Toyozumi and Kara [21] presented an analysis of a predatory vaccination application called Predator. They employed the biologically inspired Lotka-Volterra equation [22] to model the interaction of the predator-prey relationship between the malicious code and mobile predator vaccination, with the goal of minimizing the number of predators required to eliminate the malware threat. Although MC solutions present an effective approach for worm detection and patching, these approaches are not effective in terms of fast spreading worms.

F. Related work summary

Although, all of the above solutions provide countermeasures against network worms, none of these solutions provides an effective and efficient approach for zero day worm detection and containment in a disturbed environment. Hence a disturbed, automated worm detection, prevention and containment solution is required that will be effective against fast zero days worms.

III. Basic Methodology and Approach

The DNS is a globally distributed database that provides an IP address to domain name mapping. Almost all network traffic leaving a workstation host for another internet host, with which it has not recently communicated, requires a DNS

lookup. It is quite usual for network segments to be logically or physically separated in an enterprise network. Whyte et al. [23] divides the network into cells. According to Whyte et al. [23], the traffic generated by the propagation of fast scanning worms can be considered under the following three classifications: local to local (L2L), local to remote (L2R), and remote to local (R2L). In L2L, scanning worms target hosts within the boundaries of the enterprise network in which the source host resides. Topological worms employ this method to propagate. L2R refers to a scanning worm whose source host is within an enterprise network but which is targeting the whole internet. While in R2L propagation, scanning worms target hosts within an enterprise network from elsewhere within the internet. In this paper, our worm detection and containment scheme, detects the L2R propagation of worms, and alerts other peer networks using the Friends protocol in internal and external networks, of the detected worm event.

Ganger et. al. [10] first proposed the lack of a DNS lookup from a host as a tell-tale sign of worm scanning activity. In the case of a worm infection like Slammer [2], an infected host tries to send as many UDP datagrams as it can, in a short interval of time, without making any DNS requests. Our proposed scheme uses this behavioural signature as an indicator of scanning worm activity. Fig. 1 shows the placement of the elements of our resource limiting plus look ahead (RL+LA) scheme in an enterprise network. The RL+LA prototype is deployed on the internal network gateways and on the DMZ gateway to implement rate limiting and send internal Friends messages and on the border gateway of network in order to send Friends messages to external Friends peers on the internet. Each host in any network cell is allowed to send up to N outbound TCP SYN or UDP datagrams without a corresponding DNS lookup in a unit interval of time. If a host sends more than a threshold value, N, outbound UDP datagrams without appropriate DNS lookups in a specified time interval, the RL+LA implementation flags this as a worm infection indicator, use iptables to block further datagrams from the host from exiting the network cell, reduces the threshold to N/2 and sends an alert message to internal and external peers using the Friends protocol. On receipt of such a message, each peer will reduce its detection threshold to N/2.

IV. RL+LA System Design

Our RL+LA proof-of-concept implementation is coded in the C programming language. It uses the libpcap [24] library (for traffic capture) and the libpjlib [25] library (for parsing incoming DNS replies). Fig. 2 shows the flowchart of the RL+LA algorithm. For any TCP SYN or UDP datagram leaving the network, RL+LA looks for a corresponding DNS lookup in table A of Fig. 2, in the absence of a corresponding entry, it adds the source IP address to table B of Fig. 2 and increments the count value. The result of all DNS lookups along with the corresponding source and destination IP addresses is saved in table A. Table A serves as a DNS cache for all of the external hosts accessed by a host inside the network cell. Different threshold values can be defined for different networks, depending on the nature of the typical traffic of that network. . Another time interval, k, is defined in table B to decrement the values in table B. The higher the rate

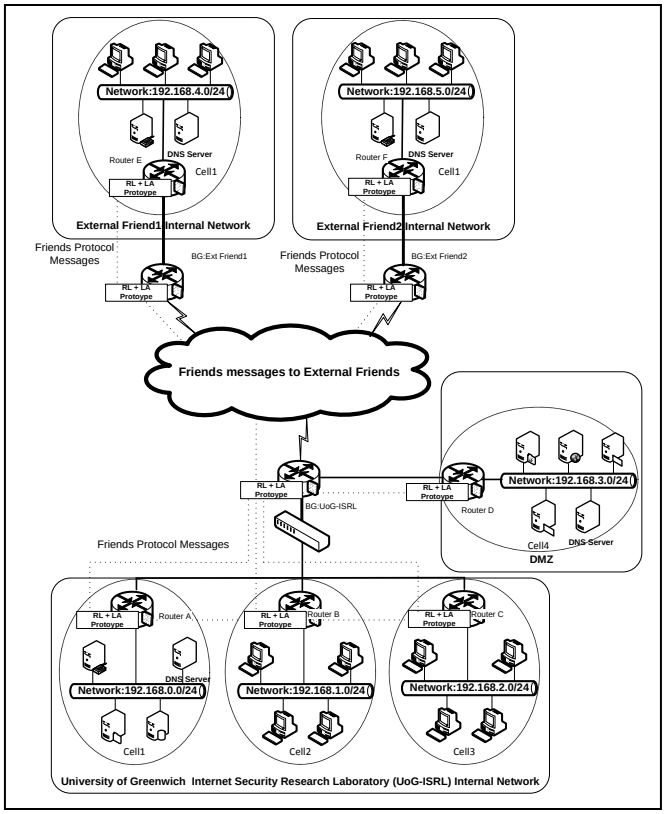

Fig. 1. Experimental Test Network

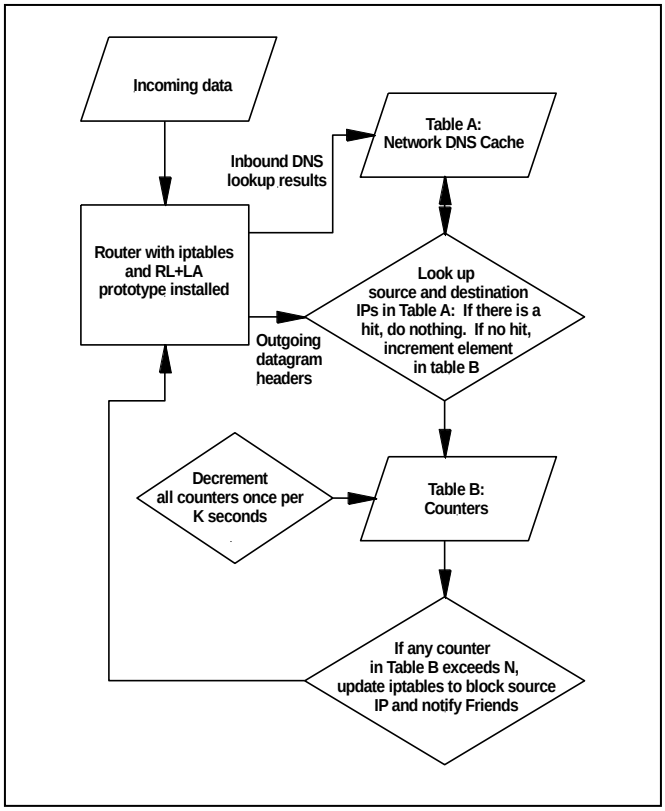

Fig. 2. Flow Chart for Prototype RL+LA Algorithm

of decrementing the values in table B, the lower the probability of a false positive being triggered. It should also be noted that some legitimate network services generate UDP datagrams without a preceding DNS lookup, such as the DNS service itself, which obtains the IP address of the DNS servers by other means at boot time. In order to address this situation, a small number of destination IP addresses are whitelisted in the system, such as those for the primary and secondary DNS servers, and other can be added.

Once the threshold value is reached in table B the RL+LA application blocks outgoing traffic from the offending host using iptables, reduces N to N/2 and sends an alert message using the Friends protocol to internal peer routers and to the border gateway, which in turn forwards the alert to external peers in the scheme, again using the Friends protocol. Each alert message contains the router user name, a predefined password, and a command to half the threshold value in table B.

V. PROTOTYPE EVALUATION

A. Network Architucture

To validate our RL+LA prototype, we set up a test bed consisting of six fully routable class C networks (192.168.0.0 to 192.168.5.0) as shown in Fig. 1. This test bed was implemented using virtualized hosts, with the underlying hardware consisting of an i7 based host with 24 GB of RAM for each class C network and a number of other hosts to implement the routers, etc. Each i7 host had VMware Exsi installed as its host operating systems. The gateway for each network ran a Linux 2.6 kernel along with iptables, the Quagga routing package [26] and the RL+LA software. Each i7 host ran 254 virtual machines, each running a Linux 2.6 kernel. The gateways of networks 192.168.0.0/24, 192.168.1.0/24, 192.168.2.0/24 and 192.168.168.3.0/24 ran RIP as routing protocol. Border gateway protocol (BGP) was configured between these routers. Border router BG:UoG-ISRL (172.16.0.1/24) contains lists of external scheme peers (in this case BG-Ext Friend1 (172.16.1.1/24) and BG-Ext Friend2 (172.16.2.1/24)). Routers A, B, C and D exchange Friends protocol alert messages directly whereas BG-ISRL forwards alert messages to external scheme peers. We have selected a test bed with 6 class C network for performing the experiment due to limitation of resources.

B. Slammer-like Pseudo-worm Software

In order to test the functionality of our prototype, we have developed a network daemon which implements a Slammer-like pseudo-worm. The daemon listens on UDP port 1434 and upon receiving a datagram with an appropriate authentication string (included for safety reasons), it begins generating UDP datagrams addressed to port 1434 and to random IP addresses. The number of UDP datagrams generated, the speed of datagram generation per second and the pool from which the random destination IP addresses are chosen are configurable parameters. We have also implemented a logging server in the C programming language. At the point of "infection", the pseudo-worm daemon sends an infected time message to the logging server. All hosts in our test network are time-synchronized by using a single NTP server.

VI. EXPERIMENTAL RESULTS

We conducted nine empirical experiments using the test network and tools described. These experiments have investigated the effect of two key variables:

- The proportion of hosts in the network, which are vulnerable to infection (ie are running the pseudo-worm daemon). We have tested values of 25, 20 and 15%. These population values are selected to investigate the effectiveness of the RL+LA scheme as an initial proof-of-concept on the scale of a typical medium-sized enterprise.
- The level of countermeasure implemented. We have conducted tests with no countermeasure (i.e. to provide a base-line), with only the local rate limiting from infected hosts, and finally with both rate limiting and the alerting protocol implemented.

For all nine tests, N was set to 15 datagrams in 5 seconds, and the counter in table B of Fig. 2 was decremented every 30s (i.e. K was set to 30s). These values were selected as a compromise between achieving maximum countermeasure effects while minimizing false positives. Each experiment was started by infecting the host at ip address 192.168.0.10.

Each worm infected host is capable of generating 10 UDP datagrams in 2.5 seconds, choosing destination IP addresses in the pool of the 6 class C networks (192.168.0.0/24 to 192.168. 5.0/24). These pseudo-worm parameters were selected due to the size of the networks and the capability of the logging instruments (i.e. the logging server) to record measurements. Fig. 3 shows the results of experiments one to three, conducted with 25 % vulnerable hosts. Without any protection mechanism in place, all vulnerable hosts are infected within approximately 18 seconds. In the second experiment, with rate limiting only as the countermeasure (no alert messages between peers), 91 % (349) of vulnerable hosts are infected within approximately 17 seconds. In the third experiment, rate limiting was implemented with alert messages and 63 % (242) of vulnerable hosts were infected, again in around 17 seconds.

Fig. 4 shows the results of experiments four to six conducted with 20 % of hosts vulnerable. In this case, without any countermeasures in place, all vulnerable hosts are infected within around 18 seconds. In the second experiment, with rate limiting only as the countermeasure (no alert messages

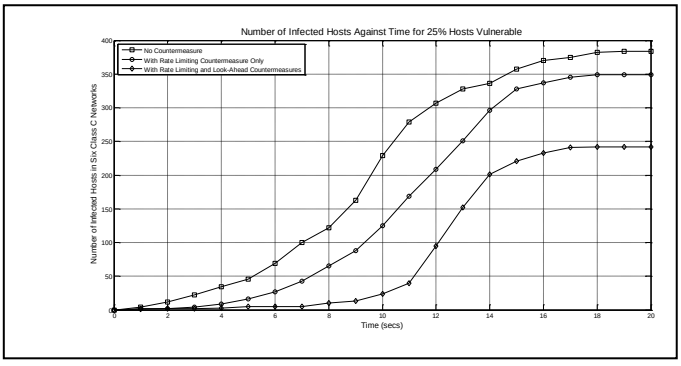

Fig. 3. Experimental Results with 25% of Hosts Vulnerable to Infection

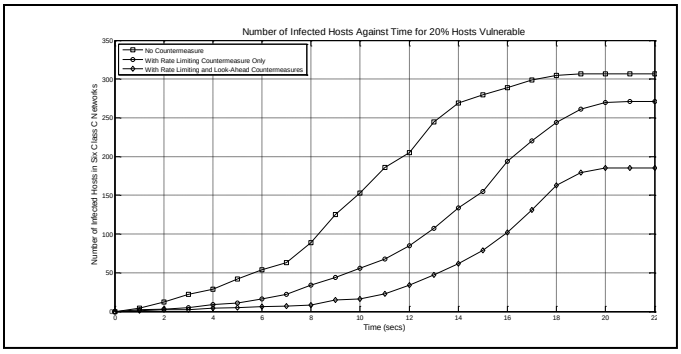

Fig. 4. Experimental Results with 20% of Hosts Vulnerable to Infection

between peers), 88 % (271) of vulnerable hosts were infected in approximately 20 seconds. In the third experiment, again, rate limiting was implemented with alert messages and, 60% (185) of vulnerable hosts were infected in around 20 seconds.

Fig. 5 shows the results of the experiments seven to nine, conducted with 15 % of the network hosts vulnerable to infection. Without any countermeasure in place, all vulnerable hosts (231 hosts) were infected within approximately 17 seconds. In the second experiment, with rate limiting only as the countermeasure (no alert messages between peers), 87.5 % (271) vulnerable hosts were infected in approximately 17 seconds. In the third experiment, again, rate limiting was implemented with alert messages and 56.5 % (131) of vulnerable hosts were infected.

VII. Analysis

Across all nine experiments, it can be seen that the RL countermeasure alone reduces the speed of propagation of the worm as well as the number of hosts ultimately infected. When the RL+LA countermeasure is employed, the speed of propagating and the number of hosts infected are further reduced. In the best case scenario (15% of hosts vulnerable to infection) the number of hosts ultimately infected is 57% of those vulnerable to the worm. This figure reduces with the percentage of hosts which are vulnerable.

In order to further quantify the effectiveness of the countermeasures, we employed the susceptible-infected SI model [27], widely used to characterize the epidemiology of biological infections and proposed by Tidy et al [28,29] and Xiang et al [30] 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 [31], r, as well as the value of β for the SI model. Figure 7 plots the SI model curve against the data from experiment three.

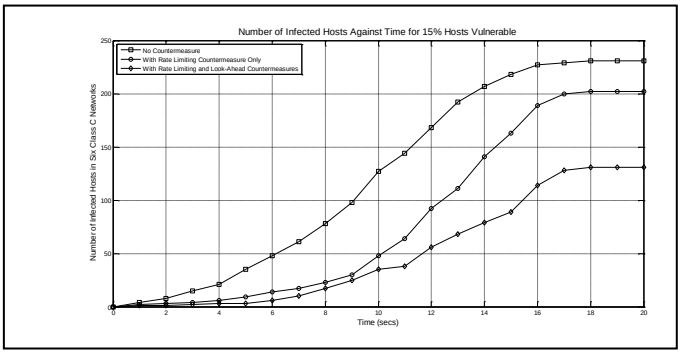

Fig. 5. Experimental Results with 15% of Hosts Vulnerable to Infection

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

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

Similar curves were fitted to each of the nine experimental data sets, and the resulting values are set out in Table I, with the relevant values of β plotted in Fig. 8.

From the data shown in Table I it can be seen that for each value of susceptible hosts tested, the value of β reduces, as first the RA, and then the RL+LA countermeasures are implemented. It is also worthy of note that the value of the correlation coefficient generally falls, as the countermeasures are implemented, indicating that the ability of the SI model to represent the experimental data reduces, as the countermeasures are invoked.

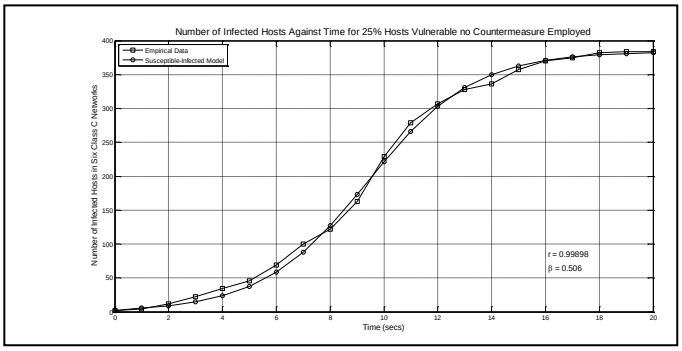

Fig. 6. SI Model Curve Fit for Experiment One Data

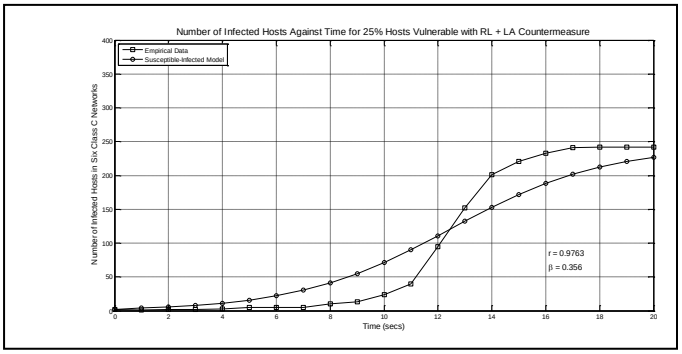

Fig. 7. SI Model Curve Fit for Experiment Three Data

TABLE I. ANALYSIS SUMMARY

Results for Tests 1 to 9				
Test Number	***Percentage of hosts susceptible***	***Counter-measure***	***β***	***r***
1	25	None	0.51	0.999
2	25	RL	0.42	0.998
3	25	RL+LA	0.36	0.976
4	20	None	0.45	0.999
5	20	RL	0.33	0.997
6	20	RL+LA	0.27	0.986
7	15	None	0.44	0.999
8	15	RL	0.34	0.994
9	15	RL+LA	0.31	0.993

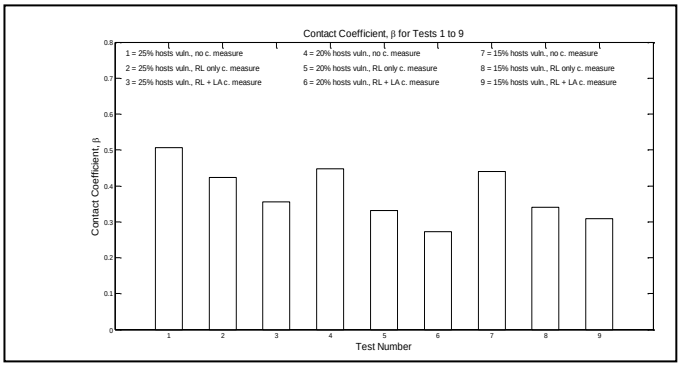

Fig. 8. SI Model Contact Coefficient Values for Each of the Nine Experiments

VIII. CONCLUSIONS AND FUTURE WORK

This paper presents a detection and containment scheme for fast scanning zero-day worms which implements a resource limiting approach based on the lack of DNS lookups along with a Friends protocol to provide leap-ahead functionality. We have developed a fully functional scheme prototype and deployed this on Linux platforms along with open source routers. Our experimental results show that the proposed scheme reduces the virulence of the tested scanning worm.

In terms of future work, DNS-lookup based detection is, itself, prone to compromise. One issue with the DNS based rate limiting mechanism is that an attacker can set up a fake external DNS server and issue a DNS query for each IP address, although adding this functionality to the worm code would significantly slow the scanning rate of the worm, as well as increasing the worm packet size. We shall, therefore be looking at ways in which such compromise can be mitigated. We shall also be exploring a wider range of variables (such as the % of vulnerable hosts) and will be investigating other behavioral signatures which can be used to detect fast scanning worms. Finally we hope to explore the performance of the proposed scheme for a small range of other worm types, and in the presence of a range of types of background traffic.

REFERENCES

[1] T. M. Chen and J.-M. Robert, "Worm epidemics in high-speed networks" in IEEE Computer 37, ppp 48–53, June 2004.

[2] 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.

[3] 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.

[4] D. Moore, C. Shannon, G. Voelker, and S. Savage, "Internet quarantine: Requirements for containing self-propagating code" in Proceedings of the 2003 IEEE Infocom Conference, San Francisco, CA, April 2003.

[5] P. Porras, L. Briesemeister, K. Levitt, J. Rowe, and Y.-C. A. Ting, "A hybrid quarantine defense". In Proceedings of the 2nd ACM Workshop on Rapid Malcode (WORM), pages 73-82, October 2004.

[6] M. M. Williamson, "Throttling viruses: Restricting propagation to defeat malicious mobile code", ACSAC, December 2002.

[7] S. Chen and Y. Tang, "Slowing down internet worms", in Proceedings of the 24th International Conference on Distributed Computing and Systems (ICDCS '2004), Tokyo, Japan, March 2004.

[8] S. Schechter, J. Jung and A. Berger, "Fast detection of scanning worm infections", in Seventh International Symposium on Recent Advances in Intrusion Detection (RAID), Sophia Antipolis, France, September 2004.

[9] M. Gualtieri and D. Mosse, "Limiting worms via QoS degradation" University of Pittsburgh, 2003.

[10] G. Ganger, G. Economou and S. Bielski, "Self-securing network interfaces: what, why, and how," Carnegie Mellon University Technical Report, CMU-CS-02-144, August 2002.

[11] C. Wong, S. Bielski, A. Studer, C. Wang, "Empirical analysis of rate limiting mechanisms". In Recent Advances In Intrusion Detection (RAID), Seattle, WA, September 2005.

[12] D. Nojiri, J. Rowe and K. Levitt, "Cooperative response strategies for large scale attack mitigation," in Proceedings of the 3rd DARPA Information Survivability Conference and Exposition, April 2003.

[13] K. Anagnostakis, M. Greenwald, S. Ioannidis, A. Keromytis and D. Li, "A cooperative immunization system for an untrusting internet," in Proceedings of the 11th IEEE International Conference on Networks (ICON), Sydney, Australia, September 2003.

[14] L. Briesemeister, P. Lincoln and P. Porras. "Epidemic profiles and defense of scale-free networks," in Proceedings of the ACM Workshop on Rapid Malcode, Washington, DC, October 2003.

[15] S. Gorman, R. Kulkarni, L. Schintler and R. Stough, "Least effort strategies for cybersecurity," Technical Report, George Mason University, 2003.

[16] S. Antonatos, P. Akritidis, E. P. Markatos and K. G. Anagnostakis, "Defending against hitlist worms using network address space randomization", in Proceedings of the 2005 ACM Workshop on Rapid Malcode (WORM), pages 30–40, 2005.

[17] N. Provos, "A virtual honeypot framework," in Proceedings of the 12th USENIX Security Symposium, San Diego, California, August 2004.

[18] J. Riordan, D. Zamboni and Y. Duponchel, "Building and deploying billy goat, a worm detection system", in Proceedings of the 18th Annual FIRST Conference, 2006.

[19] H. Kim and B. Karp, "Autograph: Toward automated, distributed worm signature detection", in Proceedings of 13th USENIX Security Symposium, August 2004.

[20] S. Singh, C. Estan, G. Varghese and S. Savage. "Automated worm fingerprinting", in Proceedings of the 6th ACM/USENIX Symposium on Operating System Design and Implementation (OSDI), December 2004.

[21] H. Toyozumi and A. Kara, "Predators: Good mobile code combat against computer viruses", New Security Paradigms Workshop, Virginia Beach, Virginia, September 2002.

[22] A.J. Lotka, Elements of physical biology, Williams & Wilkins Co, 1925.

[23] D. Whyte, E. Kranakis and P. van Oorschot, “DNS-based detection of scanning worms in an enterprise network”. In Network and Distributed Systems Symposium (NDSS), 2005.

[24] Tcpdump and libpcap public repository. http://www.tcpdump.org, accessed Feb 2014.

[25] PJLIB Library. http://www.pjsip.org/pjlib/docs/html/, accessed Feb 2014.

[26] Quagga Router Suite. http://www.nongnu.org/quagga/, accessed Feb 2014.

[27] W. Kermack and A. McKendrick, “A contribution to the mathematicaltheory of epidemics,” in Proceedings of the Royal Society of London, vol. A, no 115, pp 700–721, 1927.

[28] L. Tidy, “Investigation of zero-day worms with the aid of large-scale, high fidelity simulation”, PhD Thesis, University of Greenwich, 2014.

[29] L. Tidy, S. Woodhead and J. Wetherall, “Simulation of zero-day worm epidemiology in the dynamic, hetrogeneous internet, in Journal of Defence Modelling and Simulation. Published on-line October 2013.

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

[31] K. Pearson, “Notes on regression and inheritence in the case of two parents, in Proceedings of the Royal Society of London, no 58, pp 240-242, 1895.