Worm Containment with Dummy Addresses and Connection Trace Back

Taro Inaba[†] , Nobutaka Kawaguchi[†] , Shinya Tahara[†] ,
Hiroshi Shigeno[‡] and Kenichi Okada[‡]

Most of existing network worms have used address scanning to find vulnerable hosts. Recently, however, worms with more effective propagation strategies have emerged. Among the worms, we focus on the worms that exploit address lists obtained from infected hosts to find other vulnerable hosts effectively. In this paper, we propose a method to detect and contain such worms that try to infect all hosts in an enterprise network. In our method, a detection system inserts some dummy addresses into the address lists of hosts in the network. Then, the system detects the existence of worms when a host tries to open a connection to a dummy address, and then traces back the connection logs to find potential infected hosts and removes them from the network. Computer simulation results showed our method contained worms with less than 1% infected hosts and less than 5% removed hosts.

1 Introduction

Resent years, many worms such as CodeRed and Slammer have caused significant damages to many networks [1] [2]. Most of these worms have used address scanning to find vulnerable hosts. In address scans, an infected host generates a list of random addresses and tries to infect hosts in the list. So scanning worms open connections at very high speed and may connect to nonexistent hosts. Therefore, the strategy is inefficient and makes network conditions anomaly. Thus, by exploiting such features, many methods can contain scanning worms [4] [5].

Recently, however, worms with more effective propagation strategies have emerged [7] [8]. Among the worms, we focus on the worms that exploit address lists obtained from infected hosts to find victims [7]. By using these lists, worms surely can connect to existing hosts. Therefore, infected hosts can steadily increase and the speed of infection can be slower. It is difficult to detect and contain such worms by anomaly based methods because the worms don't make network conditions anomaly.

In this paper, we propose a method that contains these worms by using dummy addresses and connection trace back. In our method, Address Monitoring Server (AMS) monitors a whole enterprise network and adds some dummy addresses to hosts in the network. If a host tries to connect to a dummy address, AMS can detect the existence of a worm and judges the host is infected. Then AMS removes the host from the network to prevent further propagation. Moreover, AMS has connection logs of the network and traces back the logs from the infected host to find and remove potential infected hosts. Each time a host connects to dummy addresses, AMS conducts the process to remove all infected hosts from the network. To the best of our knowledge, this paper is the first work that presents a network based containment method of such worms.

The rest of this paper is organized as follows. Section 2 introduces the worm propagation strategies and related works. Section 3 proposes a worm containing method with dummy addresses and connection trace back. Section 4 evaluates our method by computer simulation based experiments. We conclude this paper in section 5.

2 Background

2.1 Scanning Worms

Existing worms such as CodeRed find vulnerable hosts by address scanning [1] [2] [3]. Scanning worms try to open so many connections that the worms make the network condition anomaly. So by focusing on this anomaly condition, scanning worms can be detected and contained [5] [6].

Matthew M. Williamson proposed Throttling Viruses to restrict scanning worms [4]. This method limits the rate of connections to hosts, with which the source hosts have not communicated, to restrict worms' propagation. Although normal traffic can be affected by this method, in reality, most of benign hosts tend to open connections to peers they have contacted with recently, and therefore only a few of connections will be used for the first contacts with new peers. Therefore, this method can restrict worms without the significant detrimental effect on normal traffic.

2.2 Worms Exploiting Internal Address Lists

If worms exploit internal address lists obtained from already infected hosts [7] [8], they can connect to surely existing hosts and can propagate slowly. Methods against scanning worm cannot detect such

[†] Graduate School of Science and Technology, Keio University
[‡] Faculty of Science and Technology, Keio University

slow worms. Therefore, new methods against worms exploiting internal address lists are required.

Chin-Tser Huang et al. proposed a method using a dummy address to detect E-mail worms [9]. In this method, a detection server inserts a dummy address into users' address lists of their E-mail software. If a host sends an E-mail to the dummy address, the sender host and the signature of the mail contents are added to a black list. After that, the detection server uses the black list to restrict E-mail worms that have an identical signature. Since the method relies on signatures of worms for containment, however, it is ineffective against polymorphic worms and encrypted worms since they can change their signatures freely.

3 Proposal

3.1 Target

The targets of our method are worms which get addresses of vulnerable hosts from internal address lists obtained from infected hosts. In this paper, internal address lists mean all address lists stored in each host machine, such as address books of E-mail, ARP caches, or connection histories to other hosts.

3.2 Contain Process

3.2.1 Overview

Figure 1 is an overview of our method.

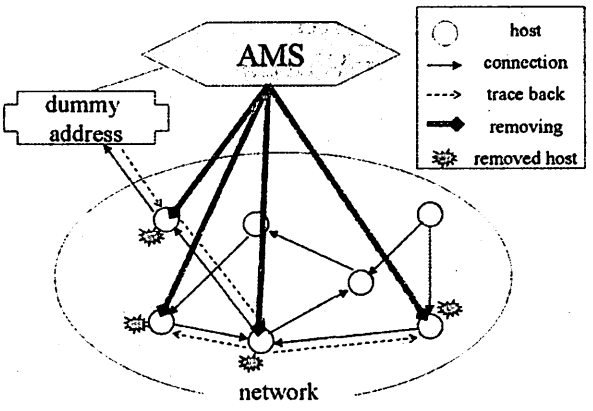


Figure 1: overview of our method

Address Monitoring Server (AMS) is a server that monitors a whole enterprise network. AMS adds some dummy addresses to each host's address lists. If a host connects to a dummy address, AMS detects the connection and regards that the source host is infected. Then AMS traces back connection logs of the network to identify potential infected hosts, and removes them from the network to prevent further propagation.

Figure 2 shows an overview of the connection trace back. In this figure, host H connects to a dummy address at t_3. Then, AMS removes the host at first. And AMS traces back from the host H to

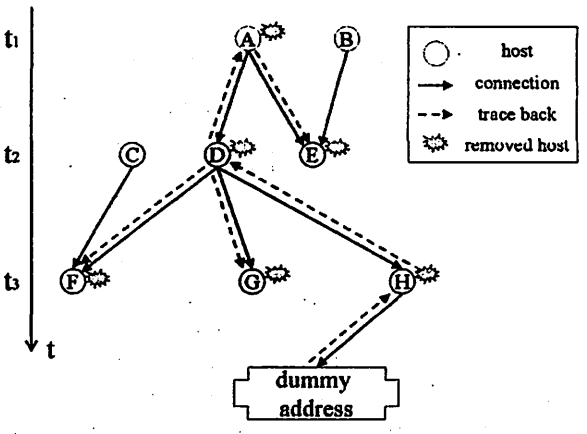


Figure 2: overview of connection trace back

find potential infected hosts and removes them. In the figure, host D, F, G, A and E are potential infected hosts and removed. This process is applied each time a new connection to a dummy address is detected in order to completely contain the worm. Since our method detects and contains worms based on only connection information, our method is also effective against polymorphic worms different from existing method [9].

The roles of AMS are as follows:

1. Monitoring all hosts in an enterprise network.
2. Adding dummy addresses to internal hosts in the network.
3. Maintaining all connection logs in the network.
4. Removing arbitrary hosts at will from the network.

In our method, all of detection and containment are done by AMS.

3.2.2 Detection

In our method, AMS detects the existence of a worm when a connection to a dummy address is opened. Because benign hosts will never connect to dummies in normal behavior, if a connection to a dummy is detected, AMS judges that the host is infected by a worm.

3.2.3 Containment by removing hosts

If a connection to a dummy address is detected, AMS removes the source host. In this section, we discuss the performance of a system that contains worms by removing the source hosts.

Now, assume a host is infected by a worm that exploits internal address lists. The infected hosts attempt to connect and infect other hosts in every Δt. When every host in the network has n valid addresses and d dummy addresses, the probability that the first infection connection is to a dummy address (P_d) is expressed as

$$P_d = \frac{d}{d+n}. \quad (1)$$

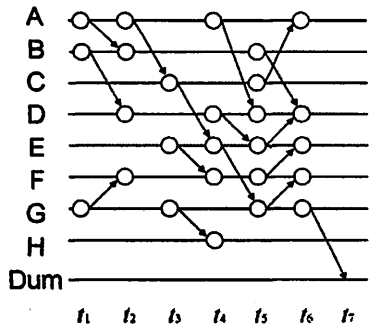


Figure 3: connection logs

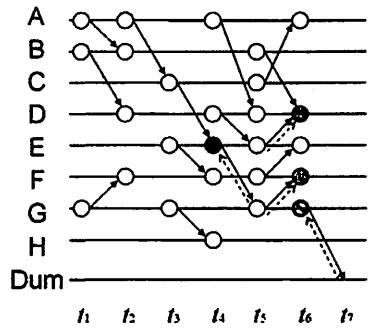


Figure 4: $N_t = 1$, $T_t = 2ut$

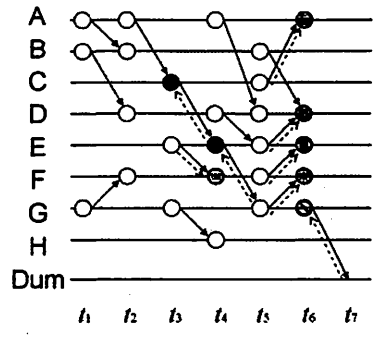
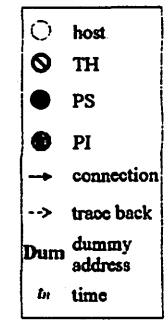


Figure 5: $N_t = 2$, $T_t = 2ut$



Therefore, the system can remove the infected host with the probability P_d. However, with the probability P_n (= 1 - P_d) the worm cannot be captured and infects new one host. Then, at next Δt, two hosts attempt to connect other hosts. The probability that both of the infected hosts connect to dummy addresses is

$$P_{d2} = \frac{d^2}{(d+n)^2} = {P_d}^2. \quad (2)$$

P_{d2} is smaller than P_d. Thus, as the number of infected hosts increases, it becomes more difficult for the system to identify and contain all the infected hosts. Therefore, it is rather impossible to contain worms by removing only the source hosts.

3.2.4 Containment by trace back

As discussed above, removing only the source hosts of connections to dummy is not enough to contain worms. To solve this problem, we introduce a connection trace back approach. Since AMS has logs of connections, AMS can trace back the logs from the source hosts. By doing this, AMS can find potential infected hosts and removes them from the network.

We describe the approach of the trace back. First, we classify potential infected hosts into the following 3 groups.

- Trigger Host (TH)
- Potential Source of Infection (PS)
- Potential Infected Host (PI)

Here, TH is a host that opens a connection to a dummy address. Therefore, TH is definitely infected. PS is a host that have connected to TH or PS. Thus, PS is suspected as a source of a worm. PI is a host which have been connected from THs, PSs, or PIs and may be infected.

Figure 3 shows connection logs of hosts. Each alphabet and horizontal line indicate a host, and Dum is a dummy address. The horizontal axis indicates time line. For the simplicity of analysis, we assume interval between t_n and t_{n+1} is $1ut$. Here, ut denotes an unit of time. In Figure 3, host G connects to a dummy address at t_6. In this case, host G is a TH and AMS traces back connection logs from G.

The trace back method has two parameters, N_t (Trace Number) and T_t (Trace Time). N_t specifies the number of steps the system traces back to find PSs, and T_t specifies the length of time AMS traces back on 1 step.

Figure 4 shows trace back with $N_t = 1$ and $T_t = 2ut$. First, since G connects to a dummy address, G is regarded as TH and removed from the network. Second, the hosts to which G has connected earlier are also suspicious because G is suspected as an infected host. As $T_t = 2ut$, the hosts G has connected to at t_4 or later are regarded as PIs and removed. In Figure 4, F is a PI. Furthermore, as $N_t = 1$, the system traces back 1 step from the TH. In Figure 4, host E, which has connected to G at t_4, is regarded as PS and removed from the network. And since the hosts to which host E connected are also suspicious, they are regarded as PIs and removed. In Figure 4, host E connects host D at t_5 and D is a PI.

Figure 5 shows another case with $N_t = 2$ and $T_t = 2ut$. In this case, AMS finds PIs and PSs by tracing back from PSs at $N_t = 1$ as well as a TH. In Figure 5, in addition to the case of Figure 4, AMS traces back from host E which is a PS with $N_t = 1$. Then, AMS removes host C as a PS and host A as a PI. If N_t is increased more, the same step will be recursively conducted.

In our proposal, AMS cannot distinguish normal connections from infection connections since we assume there is no worms signatures available. Therefore, AMS may remove potential infected hosts which are not infected in fact. Although it is difficult to prevent false detections, the number of falsely removed hosts must be minimized. We evaluate and discuss how our method minimize removed hosts in section 4.

4 Evaluation

We evaluated our method with some computer simulations written in C language.

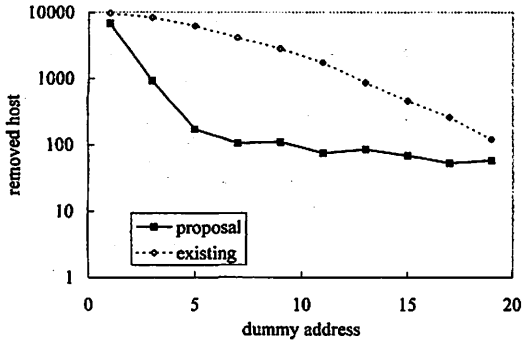


Figure 6: infected hosts

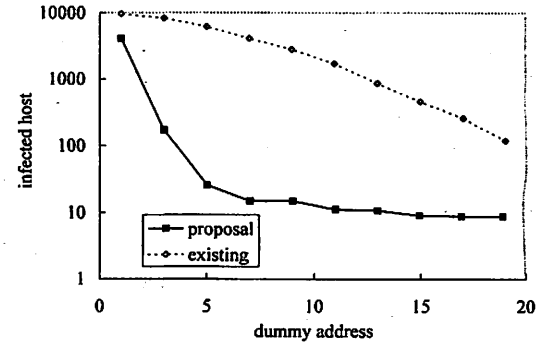


Figure 7: removed hosts

4.1 Conditions

We assume there are 10,000 hosts in the network. AMS can monitor all hosts and get connection logs in real time. At the start of this simulation, all the hosts of the network are not infected and connect to other hosts normally. At $100ut$ later, a worm appears and 5 hosts are infected. Then they begin to infect other hosts, and AMS finds and removes potential infected hosts each time a connection to a dummy address is opened. By processing this procedure repeatedly, we evaluated the number of infected hosts and removed hosts when AMS removes all active infected hosts.

Table 1: Simulation parameter

simulation times: R	30
the number of hosts: N	10000
initial infected hosts: I_0	5
vulnerable host rate: R	1.0
initial hosts at BA model: M_0	6
initial links at BA model: M	5
average addresses: A	15
dummy addresses: d	5
fraction of hosts having dummy addresses: F	1.0
trace number: N_t	2
trace time: T_t	$8ut$
normal connection interval: B_n	$5ut$
infection connection interval: B_i	$5ut$
delay of removing hosts: Q	$2ut$

We set the default simulation parameters as Table 1 and the network topology as a scale free network [10], which follows BA model [11]. In our simulation, we set the number of initial hosts (M_0) to 6 and the number of initial links (M) to 5. As a result, the average addresses (A) become 15. Since we assume relatively slow worms, the interval of continuous infection connections (B_i) and that of normal connections (B_n) have the same value, which is $5ut$. And for N_t and T_t, we set values that minimize the removed hosts when worms infection speed is equals to the speed of normal connections.

4.2 Comparison to an Existing Method

We compared our method to an existing method which uses only dummy addresses for detection and containment. We varied the number of dummy addresses, and evaluated the performance of both methods. Figure 6 and Figure 7 show the number of infected hosts and that of removed hosts respectively. Each figure's vertical axis is written in logarithmic scale. Since the optimized N_t and T_t, which minimize the number of removed hosts, depend on the number of dummy addresses (d), however, we experimented beforehand and determined the best N_t and T_t for each d in advance.

Figure 6 indicates that our method is much more effective than the existing method in the terms of the number of infected hosts. With $d = 5$, only about 30 hosts (0.3% of whole network) is infected with our method, while about 6000 hosts (60%) are infected with the existing method, when all worms are removed. The existing method can reduce infected hosts to less than 100 using about 20 dummy addresses, but ours achieves the same result with only 5 dummy addresses. Since the number of dummy addresses is preferred to be minimized, our proposed method is much better than the existing method on this point.

Figure 7 shows the number of removed hosts. With the existing method, the number of infected hosts and removed hosts are same. This is because the existing method only removes the hosts which try to connect to dummy addresses. On the other hand, with our method, the number of removed hosts is larger than that of infected hosts since AMS may falsely remove hosts that are not infected in fact by connection trace back. Nevertheless, the number of removed hosts with our method is still less than that with the existing method. In the case of $d = 5$, our method removed only 200 hosts while the existing method removed 6000 hosts before the containments are completed. Thus, from the view point of minimizing removed hosts, our method is better than the existing one.

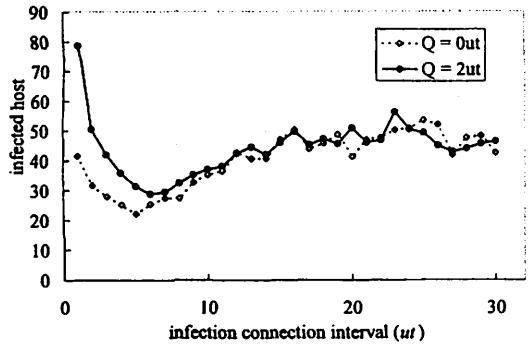


Figure 8: infected hosts: at various speed

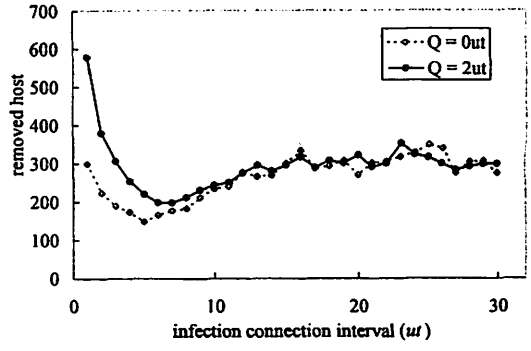


Figure 9: removed host: at various speed

4.3 Performance against Worms with Various Speed

To confirm that our method can deal with worms with various infection speed, we varied the interval of infection connection (Bi) from $1ut$ to $30ut$ and evaluated the number of infected hosts and removed hosts. Figure 8 and 9 show the results. Here Q means the delay to remove each host from the network.

As a result, with each Q, the number of infected hosts was less than 100 (1%) and that of removed hosts was less than 500 (5%) against every infectious connection speed except the case of $B_i = 1ut$ and $Q = 2ut$. Against high speed worms, such as $B_i = 1ut$ or $2ut$, the difference between $Q = 2ut$ and $Q = 0ut$ is large, but against slower worms, the difference is small. This is because infection connection interval is much longer than Q. Therefore, if AMS can keep Q much shorter than infection connection interval, our system can contain worms with various speed.

Now, we discuss the relation between speed of worms and performances of our method. In the figures, the performances are best at about $B_i = 5ut$. This is because we set the parameters that minimize removed hosts at $B_i = 5ut$. Against faster worms with smaller B_i, the performances are degraded.

On the other hand, however, against slower worms with $B_i > 15ut$, the performance are relatively constant. The result is due to the features of worms that exploit internal address lists. Hit List Worms, which include worms that exploit internal address lists, can only infect hosts that are included in the address list of already infected hosts.

If the interval of infection connections is longer than T_t, AMS cannot trace back most of causal connections, which are actually used to infect other hosts. Nevertheless, AMS can trace back some normal connections. In our simulation, we assume every host opens normal connections to the hosts in its address list at random. Here, assume every host have A normal addresses. If T_t is twice as large as the normal connection interval (B_n), there are 2 normal connections from 1 host on an average. And to remove the causal host, AMS can utilize not only a connection from the causal host to a TH but also a connection with opposite direction. Thus, when a infected host opens a connection to a dummy address, the probability that the causal host is removed (P_c) is

$$P_c = 1 - \left(\frac{A-1}{A}\right)^4. \quad (3)$$

Nc

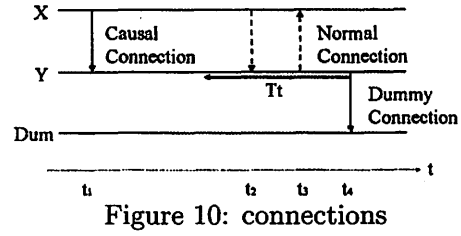


Figure 10: connections

In this figure, the host X infects the host Y at t_1 and Y connects to a dummy address at t_4. Since T_t is shorter than $t_4 - t_1$, AMS cannot trace back the causal connection t_1. However, if a normal connection between X and Y is opened within T_t from t_4, AMS can trace back this connection and remove the host X as well as Y. Like this, even the case where a causal connection cannot be traced back, it is possible to remove the source host by tracing back normal connections if A is enough small. As a result of this process, AMS can stop worms that propagate at slow speed when each host has relatively a few addresses.

4.4 Performance against Fraction of Hosts Having Dummy Addresses

In this paper, we assume all hosts cooperate to AMS and add dummy addresses to their address lists. In this section, we assume another case where some hosts do not add dummy addresses due to the various reasons. Here, we evaluate the performance of our method with the case where only 80% hosts insert dummy addresses with the parameters of list.1. Figure 11 and 12 show the result compared to the case where 100% hosts have dummy addresses.

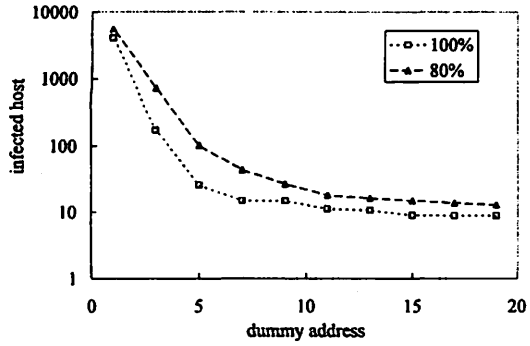


Figure 11: infected hosts: 80% hosts have dummy

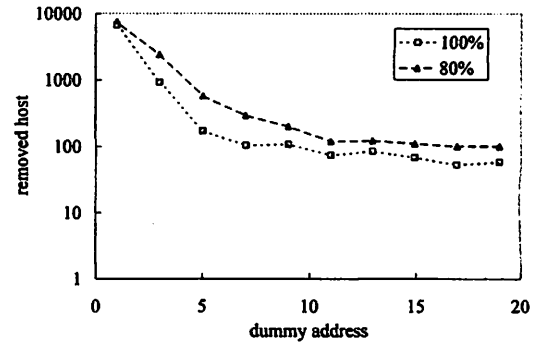


Figure 12: removed host: 80% hosts have dummy

In these results, the performance with 80% degrades compared to 100%. With $d = 5$, infected hosts and removed hosts with 80% are about 4 times as many as those with 100%. To get a similar performance to the case with 100% and $d = 5$, each host must have 9 dummy addresses when the fraction of hosts having dummy addresses is 80%. Therefore, the fraction of hosts having dummy addresses has significant impact on our performance that it is important for AMS to insert dummy addresses to all of monitored hosts surely. If AMS cannot insert dummy addresses to all the hosts, the number of dummy addresses per each host need to be increased.

5 Conclusion

These years, worms which do not use address scans to find victim hosts have emerged, and it becomes necessary to establish a method that can contain such worms. Among these worms, we focus on the worms which exploit address lists at already infected hosts and proposed a method that contain worms by combing dummy addresses and connection trace back. In our method, if a host connects to a dummy address, the system removes not only the trigger host but also potential infected hosts which are identified by trace back from the host. Using this method, worms could be contained when less than 1% hosts were infected and less than 5% hosts were removed.

In the future work, we will conduct more detailed simulations for practical use, and implement a prototype and try it in the real network environments.

Acknowledgment

This work is supported in part by a special grant from Secom Science and Technology Foundation, COE "Optical and Electronic Device Technology for Access Network" and the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research(C),2006,1850063.

References

[1] Cliff Changchun Zou, Weibo Gong and Don Towsley. Code Red Worm Propagation Modeling and Analysis. Proceedings of the 9th ACM conference on Computer and communications security, 2002.

[2] CERT Advisory CA-2001-26 Nimda Worm http://www.cert.org/advisories/CA-2001-26.html, September 2001.

[3] Code Red II: Another Worm Exploiting Buffer Overflow in IIS Indexing Service DLL http://www.cert.org/incident_notes/IN-2001-09.html, August 2001.

[4] Matthew M. Williamson. Throttling Viruses: Restricting propagation to defeat malicious mobile code. Proceedings of Computer Security Applications Conference, 2002.

[5] Stuart E.Schechter, Jaeyeon Jung, and Arthur W. Berger. Fast Detection of Scanning Worm Infections. Recent Advances in Intrusion Detection, 2004.

[6] David Whyte, P.C. van Oorschot, Evangelos Kranakis. ARP-based Detection of Scanning Worms Within an Enterprise Network. Carleton Univ Technical Report, 2005.

[7] Nicholas Weaver, Vern Paxson, Stuart Staniford, Robert Cunningham. A Taxonomy of Computer Worms. Proceedings of the 2003 ACM workshop on Rapid malcode, 2003.

[8] http://www.symantec.com/avcenter/venc/data/w32.bropia.html, May 2007

[9] Chin-Tser Huang, Nathan L. Jhonson, Jeff Janies, Alex X. Liu. On Capturing and Containing E-mail Worms. Proceedings of IEEE IPCCC 2006, 2006.

[10] Cliff C. Zou, DonTowsley, Weibo Gong. Email Worm Modeling and Defense. 13th International Conference on Computer Communications and Networks, 2004.

[11] Sinya Ishida, Shin-ich Arakawa, Masayuki Murata. Power-Law Property in WDM Networks: Does the Wavelength Conversion or Wavelength Routing resolve it? TECHNICAL REPORT OF IEICE, December 2003.