# Detecting Unknown Worms using Randomness Check *

Hyundo Park and Heejo Lee **

Korea University, Seoul 136-713, South KOREA
{hyundo95, heejo}@korea.ac.kr

**Abstract.** From the appearance of CodeRed and SQL Slammer worm, we have learned that the early detection of worm epidemics is important to reduce the damage caused by their outbreak. One prominent characteristic of Internet worms is to choose next targets randomly by using a random generator. In this paper, we propose a new worm detection mechanism by checking the random distribution of destination addresses. Our mechanism generates the traffic matrix and checks the value of rank of it to detect the spreading of Internet worms. From the fact that a random binary matrix holds a high value of rank, ADUR (Anomaly Detection Using Randomness check) is proposed for detecting unknown worms based on the rank of the traffic matrix. From the experiments on various environments, we show that the ADUR mechanism effectively detects the spread of new worms in an early stage, even when there is only one host infected in a monitoring network.

## 1 Introduction

An Internet worm is the one of malicious codes that propagates by copying itself onto other computers. Such a self-replicating malicious code scans vulnerable hosts on a network, replicates itself to the vulnerable hosts without user intervention. Starting from one decade after the Morris worm in 1988, the number of incidents according to Internet worms is growing drastically. CodeRed and Nimda worm infected hundreds of thousands of vulnerable computers at year 2001. At that time, from public institutes to personal users suffered from the damage caused by CodeRed and Nimda. The amount of damage was accounted for millions of dollar [1-4].

On the history of Internet worms, the SQL Slammer worm is known as the fastest spreading worm. It spends mere 10 minutes to infect 90 percent of vulnerable hosts in the Internet. And the number of infected hosts increased two times per 8.5 seconds. This speed is much faster than CodeRed, which increases two times per 37 minutes [5]. The first step toward countering with the epidemic of a worm is ”early detection” as soon as possible. However, signature-based detection algorithms are not proper to detect new worms or polymorphic worms since they can always change their codes. On the contrary, anomaly-based approaches can be used for detecting such worms, at the expense of higher degree of complexity and false alarms.

In this paper, we propose a new method for detecting the spread of Internet worms, which is called ADUR (Anomaly Detection Using Randomness check). The ADUR

* This work was supported in part by the ITRC program of the Korea Ministry of Information & Communications.

** To whom all correspondence should be addressed.

mechanism can detect a new worm by measuring the randomness of destination addresses in network traffic, where the randomness is formed when a worm propagates randomly over the Internet. By checking the randomness of address distribution, ADUR distinguishes between the status of normal conditions and the state of worm epidemics. Two main features of ADUR are the use of "matrix" representations and exclusive-or (XOR) operations. Matrix representations give many benefits to implement particular operations regardless of network size and traffic volume. And the XOR operator diminishes the effect of normal traffic and magnifies the effect of worm traffic, which results in reduced false alarms. By measuring the dynamics of the rank of traffic matrix, ADUR can detect the worm epidemic in an early stage.

Main contributions of this study is three-fold. First, we propose a novel approach to detect unknown worms based on the randomness of worm traffic. Second, we suggest an anomaly detection algorithm based on matrix and its simple operation of XOR, which greatly increase the flexibility and the accuracy of detection. Finally, the algorithm gives additional information such as infected subnet locations when a worm is detected.

The rest of this paper is organized as follows. In Section 2, we discuss previous research on scan detection, and related work. In Section 3, we explore scanning methods of Internet worms which have been used to choose a target host. Section 4 describes how to check the randomness of traffic. In Section 5, we propose the ADUR mechanism and show the reason why it uses the XOR operator. The evaluation of the proposed ADUR mechanism is shown in Section 6. We summarize our results and conclude the paper in Section 7.

## 2 Related Work

In general, the algorithms to detect worm are divide into two classes, signature based and anomaly based detection algorithm. The signature based worm detection algorithm uses pattern matching method. And anomaly based worm detection algorithm uses characteristic of worms.

The signature based worm detection algorithm is widely used on the most of the system to detect worm. The first this system gathers the information of specific worm after the worm is propagated. And then, this system make the signature of the specific worm. So, this system uses it to detect known worm. For example, the CodeRed worm make the pattern composed the ASCII value on the network traffic. Now, this value is the signature of the CodeRed worm[]. If this signature is detected on the network traffic in future, it is worm propagation state. Such algorithm has advantage which is low probability of false-positive alarm rate to known worm. But it can't detect to unknown worm. And in the case of vary fast propagation worm, such as SQL_Slammer, the signature of worm based algorithm has overhead to make the signature of it in the early stage of infection by worm. So, this algorithm is not suitable to detect worm.

Anomaly-based approaches have been studied in several ways. Zou et. al. [6] proposed a Kalman filter-based detection algorithm which detects the trend of illegitimate scans to a large unused IP space. Wu et. al. [7] proposed a victim counter-based detection algorithm that tracks the increased rate of new infected hosts. The algorithm warns when abnormal events occur consecutively over a certain number of times. Berk

[8] proposed an anomaly detection algorithm using ICMP "Destination Unreachable" messages which can be collected at border routers to infer worm activities. Previous approaches determine an on-off binary condition based on a "threshold" value, but rarely provide further information for defending the worm epidemic such as infected subnet locations in a monitoring network. Many number of anomaly-based worm detection methods are developed recently.

in the recently anomaly based worm detection researches, there is the TRW, Threshold Random Work, researching in the MIT University. The TRW is algorithm to detect worm using TCP protocol. The TRW regards a SYN packet to initial connect on the TCP protocol as a scanning method of worm. The TRW finds abnormal traffic pattern using by sequential hypothesis testing method on the basis of the information of SYN packet to initial connection on the TCP protocol. First, the TRW divides into two classes, success or fail to TCP connection. Next, if it is happened the TCP connection failure of specific host, the TRW change the value of the host. In the TRW, there exist the value per the individual host. if the TCP connection failure is continued, the value of the host increases. So, if this value is increased over the threshold, the TRW gives warning of the attack worm. the TRW is not difficult to the complexity to embody on the system. But the numerical formula of the TRW has complexity. Also, Because of the TRW detects worm by using the information of TCP connection, it can't detect worm using UDP protocol to propagate, such as SQL_Slammer. the TRW uses a memory on the system per the host which requests TCP connection. So, if the worm propagates on the internet vary fast, the connection failures many increase and the system applied to the TRW uses excessive memories. The Curious Yellow[] worm uses the distributed scanning method. In this case, if the number of connection failure of one host is not over threshold, the TRW can't detect this worm.

In this paper, we propose new anomaly worm detection mechanism using by randomness check. the ADUR mechanism is this. the ADUR can detect worm which can't detect by using the TRW at early stage of worm propagation. and the ADUR has merit using memory on the system compare with the TRW.

## 3 Scanning Methods of Active Worms

In this section, we describe the scanning methods of Internet worms and show the relationship to the randomness of traffic generated by an individual scanning method. Scanning methods can be classified onto four categories [9]: hitlist scanning, topological scanning, local scanning, and permutation scanning.

In order to show the spreading speed of Internet worms, we can use an analytical model such as AAWP (Analytical Active Worm Propagation) [15]. AAWP is a worm propagation model based on discrete time.

In the AAWP model, the number of infected hosts at time tick $i$ is shown in Eq. (1), where $N$ is the total number of vulnerable hosts in the Internet, $T$ is the size of address space used by the worm to scan, $s$ is the scan rate, and $n_i$ is the number of infected hosts at time tick $i$.

$$n_{i+1} = n_i + [N - n_i] \left(1 - \left(1 - \frac{1}{T}\right)^{sn_i}\right) \tag{1}$$

**Table 1.** Attack signatures of nine attacks

| Scanning method | Description | Example |
|---|---|---|
| Hitlist scanning | The list of vulnerable hosts is used and outgoing connections are increasing suddenly. | Warhol [10] |
| Topological scanning | The information of target is gathered on the infected host, and outgoing connections are increasing suddenly. | Morris [11] |
| Local scanning | Most targets are selected within the local network, and failed messages for connection requests are increasing. | CodeRed[12], Nimda [13] |
| Permutation scanning | This is a sort of local scanning but avoids the overlapping of scanning ranges. | Slammer [14] |

On Eq. (1), let us assume that the initial time tick is 0, i.e. i=0. And the value of n0 is equal to the initial hitlist size. Fig. 1 shows the distribution of infected hosts as a function of time tick. Even with a different hitlist size, the number of infected hosts increases drastically when the value of n passed 10,000, as shown in Fig. 1.

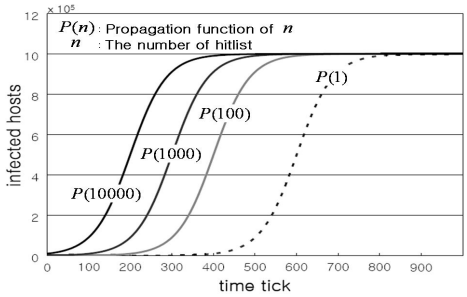


**Fig. 1.** The number of infected hosts as a function of time tick when n0 = 1, 100, 1000, and 10000, respectively.

At the beginning of worm propagation, the speed of propagation is slow. However, when the number of infected hosts exceeds a certain point, e.g. 10,000 in Fig. 1, the infection is growing rapidly. This is caused by the fact that, as the propagation proceeds, the number of scanning packets also increases along with the increased number of infected hosts. Thus, infection is accelerated by finding remaining susceptible hosts more rapidly.

A hitlist scanning worm has a list of IP addresses and the sequence of the addresses is not likely to form a uniform distribution. Thus, the traffic generated by hitlist scanning has the property of randomness. Topological scanning worms gather the information of target hosts from the information on the infected host. The sequence of target IP addresses, gathered on the infected host, is not uniformly distributed. Therefore, the traffic generated by the topological scanning worm has also the randomness. The local

and permutation scanning worm uses a random generator to generate target IP addresses with particular constraints. Eventually, the sequence of those IP addresses generated by the local or permutation scanning, have the randomness.

Thus, conventional worm propagation strategies produce the property of ”randomness” in target hosts. This implies that we can monitor the spreading of worms by measuring the randomness of destination addresses in network traffic. In this study, we attempt to measure the degree of randomness in traffic in order to catch the spreading of high speed worms.

## 4 Matrix Rank as a Randomness Metric

Many approaches have been proposed for testing the randomness and one cost-effective approach is checking the linear-dependency among fixed-length substrings of its original sequence. In order to check the linear-dependence among rows or columns of a matrix, we can use the rank of a matrix [17]. The randomness test using by the rank of the matrix has been broadly used as defined in a specification of one of tests coming from the DIEHARD [18] battery of tests.

One easy way to compute the rank of a matrix is counting the number of non-zero rows after applying the Gaussian elimination method to the matrix. In other words, the rank of the matrix is equal to the number of leading 1’s [16].

In case of a random m x n binary matrix, the value of rank has the following probability where rank r = 1, 2,. . . , min(m, n) [17].

$$2^{r<n+m-r>-nm} \prod_{i=0}^{r-1} \frac{\left(1-2^{i-n}\right)\left(1-2^{i-m}\right)}{\left(1-2^{i-r}\right)} \tag{2}$$

From the Eq. (2), we can get the distribution of probability for a given random matrix. In case of 64x64 random binary matrices, the distribution of probabilities for the value of rank is shown as Fig. 2.

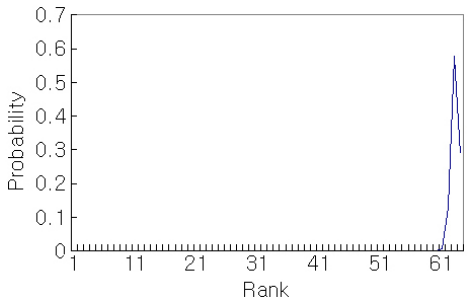


**Fig. 2.** Probability distribution of the rank of a 64x64 random binary matrix.

If one 64x64 random matrix is given, the probability that the value of rank exceeds 60 is over 99.995% from Eq. (2). This implies that, if the 64x64 binary matrix is a random matrix, the rank of the matrix will be larger than 60 with high probability. Thus, we can use the rank of a matrix to determine the randomness of element distribution.

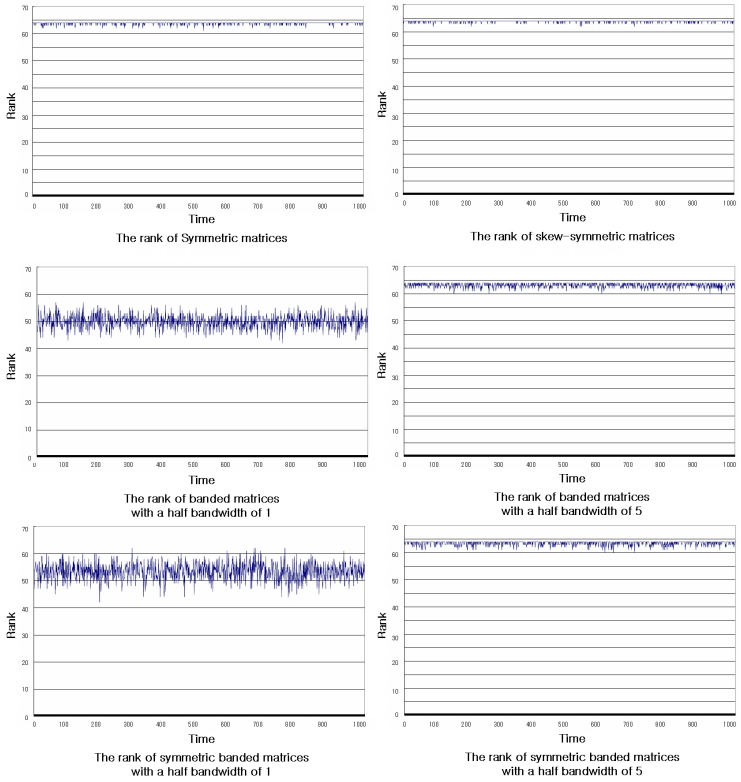


**Fig. 3.** aspects of the rank of various matrices.

## 5 Anomaly Detection using Randomness Check

We propose an anomaly-based worm detection algorithm, called ADUR (Anomaly Detection Using Randomness check). This section describes the ADUR mechanism which includes a matrix representation and its XOR operations. We show how to express network traffic on a matrix, and how to use the XOR operation in order to diminish the effect of normal traffic and magnify the effect of worm traffic. Then, the rank of the matrix is used for measuring the randomness of traffic.

### 5.1 ADUR System Design

The ADUR mechanism is to detect the spreading of Internet worms through checking the randomness of traffic. Traffic data can be classified into two categories based on their direction: incoming and outgoing. ADUR checks two directions respectively in order to get more accurate attack information such as either entering or departing the network. Checking the randomness of traffic can be accomplished by calculating the value of rank of the matrix representing network traffic for a given period of time.

Let $M_I$ denote the matrix marked with incoming traffic. And we let $M_O$ represent the matrix marked with outgoing traffic. $R(M_I)$ and $R(M_O)$ represent the value of rank of matrix $M_I$ and $M_O$, respectively. Then, the value of $R(M_I)$ and $R(M_O)$ can be used to determine whether worm is active or not. There are four states depending on the ranks $R(M_I)$ and $R(M_O)$ of traffic: calm, flowing, ebbing and flooding.

1. Calm: When both of $R(M_I)$ and $R(M_O)$ remains in a small range, there is no suspicious activity of worms.
2. Flowing: When $R(M_I)$ suddenly increases but $R(M_O)$ remains steady in a small range, the internal network is under attack by the worms in other networks.
3. Ebbing: When $R(M_I)$ remains steady in a small range but $R(M_O)$ suddenly increase, the worms in the internal network are attacking other networks.
4. Flooding: When both $R(M_I)$ and $R(M_O)$ suddenly increase, the internal network is flowing and ebbing.

### 5.2 Traffic Matrix Design

The ADUR mechanism detects the worm spreading using the value of rank of the matrix marked with network traffic. Matrix representation for both incoming and outgoing traffic is described as follows.

In IPv4, we can capture source IP address and destination IP address and divide each IP address into four octets. The length of each $IP_1$, $IP_2$, $IP_3$ and $IP_4$ is one octet.

$$IP_1.IP_2.IP_3.IP_4 \tag{3}$$

Matrix expression consists of two operations. The first one is placement. The other one is storing information. When a packet is captured in a monitoring network, the packet is mapped into a specific location of a matrix, which is determined by the destination address of the packet. Without loss of generality, we assume that the matrix size is 64x64 when the size of monitoring network is /24. Then the placement function is described as Eq. (4).

$$\begin{aligned} i &= (IP_4/16) \times 4 \\ j &= (IP_4\,(mod\,16)) \times 4 \end{aligned} \tag{4}$$

Next, we need to store some information of the packet onto the matrix. The matrix size 64x64 can hold 16 bits information for each packet. Since worms are likely to change the last two octets more frequently than the first two octets, the last two octets

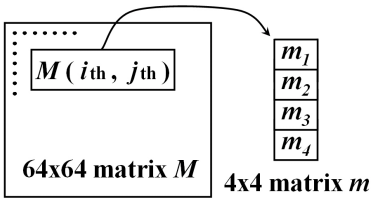


**Fig. 4.** Construction of matrix M by mapping a packet to a sub-matrix m.

of an IP address will be stored at the 4x4 sub-matrix. In case of outgoing traffic, the destination address of a packet is used for storing such information onto the matrix. In case of incoming traffic, the source address of a packet is used instead of the destination address. Fig. 3 illustrates the information storing on a 4x4 sub-matrix, where the matrix size is 64x64. Furthermore, the 4x4 sub-matrix consists of four 1x4 sub-matrices, i.e. m1, m2, m3, m4 in Fig. 3. The contents of 1x4 sub-matrices are described in Eq. (5).

$$\begin{aligned} m_1 &= first\ 4\ bit\ of\ IP_3 \\ m_2 &= last\ 4\ bit\ of\ IP_3 \\ m_3 &= first\ 4\ bit\ of\ IP_4 \\ m_4 &= last\ 4\ bit\ of\ IP_4 \end{aligned} \tag{5}$$

We can extend the 64x64 matrix representation in a /24 network to larger networks. When monitoring a /16 network, we can increase the size of matrix such as the 256 number of 64x64 matrices, i.e. 256x64x64. In this way, the principle of matrix expression can be scalable to any size of network. Reducing the matrix size for larger networks is the future work of our study.

### 5.3 XOR Operator on Matrix Sequence

We need to consider only suspicious traffic but eliminate the effect of legitimate traffic. It is found that the simple operation XOR is greatly effective for this purpose. Let $M_t$ denote the matrix at time $t$. Eq. (6) shows the XOR operation on a sequence of matrices, which dramatically reduces the influence of normal traffic on the value of rank.

$$R\,(M'_t) = R\,(M_t \oplus M_{t-1}) \tag{6}$$

Matrix $M'$ is made by the XOR operation of two consecutive matrices, which eventually remove the most portion of legitimate traffic in the matrix because legitimate traffic lives longer than one time unit. When time tick is an apt short, the legitimate traffic is sufficiently removed without the loss of the impact of suspicious traffic. Experimental results will be shown for different time ticks in next section. The XOR operation on the consecutive matrices is the key factor to detect the worm spreading.

## 6 Evaluation of ADUR

In order to evaluate the effectiveness of ADUR, we show the exponential results with real traffic. The traffic data is used the packets captured at a university network in July

29, 2004. For the purpose of making a situation of worm epidemic, we have injected various types of random scanning packets on the traffic.

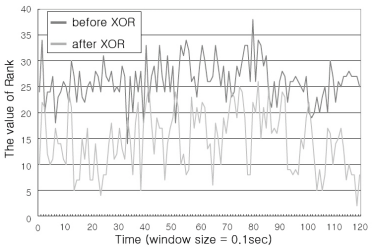


**Fig. 5.** The rank of matrix before and after XOR operation.

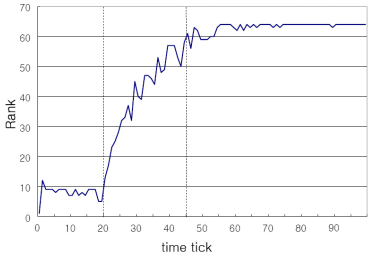


**Fig. 6.** The rank of matrix when randomly generated connections are added.

### 6.1 Effect of XOR

The ADUR mechanism eliminates the effect of legitimate connections in the traffic by the use of XOR operation on the consecutive matrices. Fig. 4 shows the values of rank ”before” and ”after” XOR operation. Since the matrix after the XOR operation has only suspicious traffic, such as new connections on the network, the value of rank greatly reduces comparing with the matrix before the operation. In other words, the ranks will increase sharply when new worms spread over the network.

### 6.2 Rank and Random Connection

Fig. 5 shows the value of rank when adding more random connections. After 20th tick, we injected one random connection per time tick. When there are more than 25 random connections, the value of rank becomes over 60. This shows the transition of state from ”calm” to ”ebbing.” From the fact that conventional worms can generate thousands of new connections per second, we can confirm that the ADUR mechanism has an ability to detect new worms in an early stage of the worm propagation.

### 6.3 Effect of Window Size on the Value of Rank

Traffic matrix M is constructed by the traffic gathered for a given time period, called ”window size,” then the value of rank is measured after passing every window size. This unit time is counted as one time tick in this paper. Here we conduct an experiment for measuring the effect of window size.

The Fig. 6 shows the rank of matrix as a function of time tick with three different window sizes. Each graph shows the result with the same traffic data in calm state. As the window size increases, the amount of traffic for construction a matrix M also

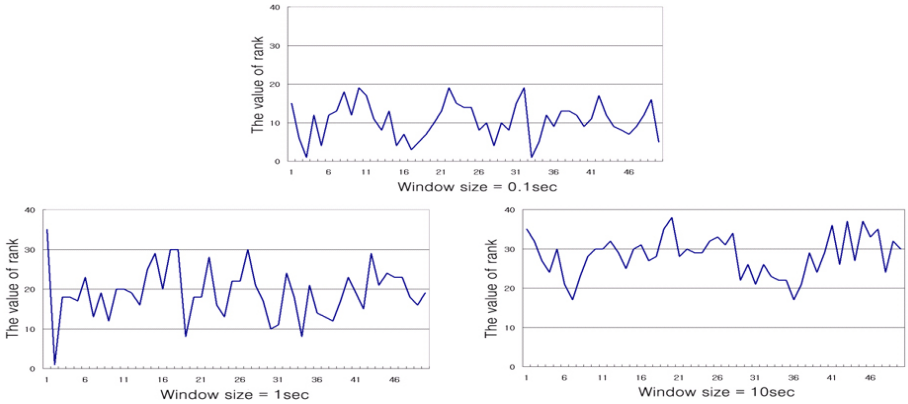


**Fig. 7.** The relation between the value of rank and the windows size.

increases. It implies that the rank of M will increase as the windows increases. However, this increment is quite limited in a certain boundary, e.g. 20 as shown in Fig. 6, but adding random connections will greatly increase the range of the rank. It shows that the ADUR mechanism is robust to the window size and also to the traffic volume. Also, it shows the possibility that ADUR can be used in high-speed networks.

In order to evaluate the effectiveness of ADUR, we measured the dynamics of ranks as the worm proceeds. Fig. 7 shows the number of infected hosts which is modeled with AAWP. Fig. 8 shows the variation of ranks associated with the propagation shown in Fig. 7. The rank of the traffic matrix by AAWP model responds quicker than the speed of infection, which is depicted by Fig. 7 and Fig. 8. In Fig. 8, two different networks

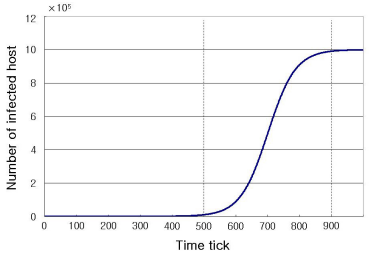


**Fig. 8.** The number of infected hosts modeled by AAWP as a function of time tick.

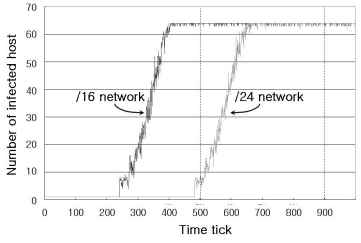


**Fig. 9.** The corresponding value of rank when worms spread with the AAWP model.

were considered: /24 and /16 as the size of a monitoring network. Monitoring larger networks can give better looking glasses so that results in earlier detection. Even in a small network such as a /24 network, we can catch the symptom rapidly such as 650 unit time in Fig. 8, where only 20% of hosts in the network got infected as shown in

Fig. 7. It implies that the ADUR mechanism is effective even by monitoring a small network. The effectiveness of ADUR is measured in a real network traffic. The traffic

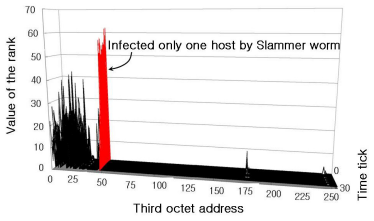


**Fig. 10.** Rank distribution for a /16 network, where only one host is infected by Slammer.

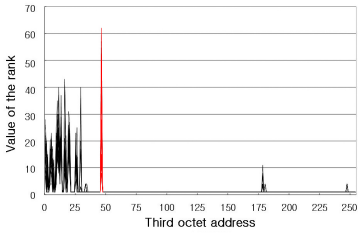


**Fig. 11.** Corresponding 2-D graph to Fig. 9, which also shows the infected subnet location.

is gathered in a university network and worm traffic is injected to the normal traffic, where only one host is infected by the Slammer worm. The rank distribution is shown in Fig. 9 and Fig. 10. This is the situation that one host located in an unused network is infected. Fig. 9 and Fig. 10 show the case that the infection of a subnet 48, which means the third octet is 48 in a /16 network. Why we input the infected host on unused local network? Because when the infected host exists in an unused network, the change of the rank by the Slammer worm correctly distinct from the normal condition. The value of rank with normal traffic is under about forty. But the value of rank with the traffic including one host infection is over sixty. So, even though only one host is infected by a worm, the ADUR mechanism can detect such infection and give additional information such as infected subnet locations.

### 6.4 Evaluation of Real Network Using by ADUR

## 7 Conclusion

We have proposed an unknown worm detection algorithm called ADUR. The proposed mechanism examines whether destination addresses have the property of random distribution. Matrix expression of network traffic and simple XOR operation on two consecutive matrices give a clue of worm spreading by measuring the rank of the matrix. This paper showed that the ADUR mechanism can detect unknown worms in an early stage of worm spreading and robust to the size and speed of a monitoring network and the volume of traffic. We have a plan to run this mechanism in high speed networks and try to experiment with more instances. Furthermore, we will extend the mechanism to work when multiple worms spread simultaneously.

## References

1. R. Russell and A. Machie: CodeRed II worm, Tech. Rep., Incident Analysis, Security Focus, Aug. 2001.

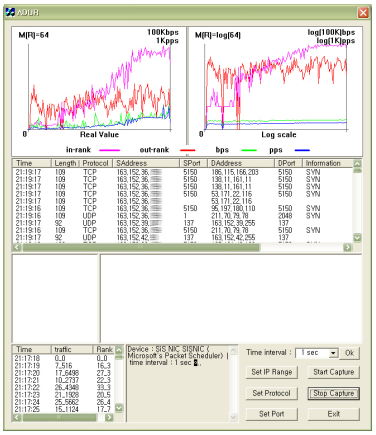


**Fig. 12.** ADUR application.

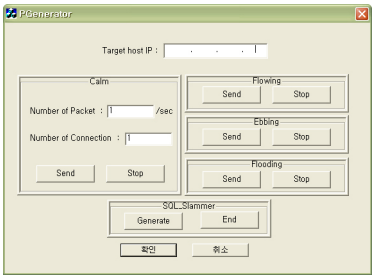


**Fig. 13.** ADUR application.


2. A. Machie, J. Roculan, R. Russell, and M. V. Velsen: Nimda worm analysis, Tech. Rep., Incident Analysis, SecurityFocus, Sep. 2001.
3. CERT/CC: CERT Advisory CA-2001-26 Nimda Worm, http://www.cert.org/advisory/CA-2001-26.html, Sep. 2001.
4. D. Song, R. Malan, and R. Stone: A snapshot of global Internet worm activity, Tech. Rep., Arbor Network, Nov, 2001.
5. H. Park and H. Lee: Evaluation of malicious codes, Tech. Rep., IIRTIRC, 2004.
6. C. C. Zou, L. Gao, W. Gong, and D. Towsley: Monitoring and early warning for Internet worms, Proc. of ACM CCS, Oct. 2003.
7. J. Wu, S. Vangala, L. Gao, and K. Kwiat: An efficient architecture and algorithm for detecting worms with various scan techniques, Proc. of NDSS, Feb. 2004.
8. V.H. Berk, R.S.Gray, and G. Bakos: Flowscan: Using sensor networks and data fusion for early detection of active worms, SPIE AeroSense, Vol. 5071, pp. 92-104, 2003.
9. S. Sraniford, V. Paxson, and N, Weaver: How to own the Internet in your spare time, the 11th USENIX Security Symposium (Security '02), Aug. 2002.
10. N. Weaver: Warhol worms: The potential for very fast Internet plaques, http://www.cs.berkely.edu/˜nweaver/warhol.html.

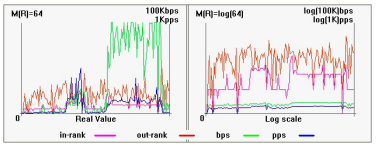


**Fig. 14.** normal state.

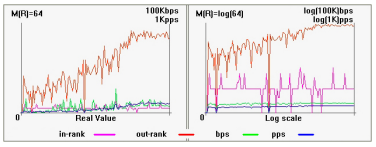


**Fig. 15.** flowing state.


11. M. Eichin and J. Rochlis: With microscope and tweezers: An analysis of the Internet virus of November 1988, IEEE Symposium on Security and Privacy, 1989.
12. C. C. Zou, W. Gong, and D. Towsley: CodeRed worm propagation modeling and analysis, Proc. of ACM CCS, Nov. 2002.
13. A. Machie, J. Roculan, R. Russell, and M. V. Velzen: Nimda worm analysis, Tech. Rep., Incident Analysis, SecurityFocus, Sept. 2001.
14. D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford and N. Weaver: Inside the slammer worm, IEEE Magazine of Security and Privacy, pp. 33-39, Jul./Aug. 2003.
15. Z. Chen, L. Gao, K. Kwiat: Modeling the spread of active worms, IEEE INFOCOM, 2003.
16. H. Anton: Elementary linear algebra, 7th ed. John Wiley & Sons, Inc. 1994.
17. G. Marsaglia and L. H. Tsay: Matrices and the structure of random number sequences, Linear algebra and its applications 67, pp. 147-156, 1985.
18. G. Marsaglia: DIEHARD: a battery of tests of randomness, http://stat.fsu.edu/˜geo/diehard.html.


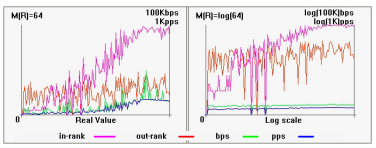


**Fig. 16.** ebbing state.

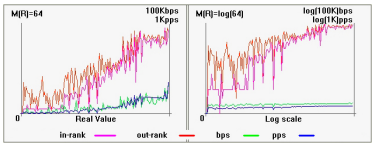


**Fig. 17.** flooding state.

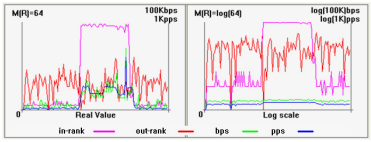


**Fig. 18.** infected by slammer worm.