Front. Electr. Electron. Eng. China 2007, 2(4): 464–472
DOI 10.1007/s11460-007-0087-7

RESEARCH ARTICLE

CHEN Bo, YU Xiangzhan, FANG Binxing, YUN Xiaochun

Worm attack detection and response

Abstract There appear many Internet-scale worm incidents in recent years, which have caused severe damage to the society. It is clear that a simple self-propagation worm can quickly spread across the Internet. Therefore, it is necessary to implement automatic mitigation which can detect worm and drop its packet. In this paper, the worm's framework was first analyzed and its two characteristics were detected. Based on the two characteristics, a defending algorithm was presented to protect network. Experimental results verify that our algorithm is very effective to constrain the worm propagation and meanwhile it almost does not interfere in normal activity.

Keywords worm detection, network monitoring, connection degree, lossy counting

1 Introduction

Nowadays, more and more persons log onto Internet for recreating, downloading files or just browsing web pages. The Internet thus plays an important role in the economy of a country besides people's life. Once the internet is infected by worms, it will break down, which consequently results in enormous economic loss. Worms will cause network congestion that poses severe threat on Internet. In 1988, Moriis worm struck the world and more than 6 000 servers were infected and the network paralyzed for a few days. It directly cost more than ten million dollars of economic losses. On July 19, 2001, the Code Red Worm breakout launched a new tide of Internet-scale worm attacks. After that, Code Red II, Nimda, Slammer, Blaster, Sasser and Witty attacked the Internet time after time and caused great damage to the society. In this sense, people and organizations began to pay much more attention to detecting and defending against worms.

Translated from *Journal on Communications*, 2007, 28(2): 9–16 [译自: 通信学报]

CHEN Bo (✉), YU Xiangzhan, FANG Binxing, YUN Xiaochun
Research Center of Computer Network and Information Security Technology, Harbin Institute of Technology, Harbin 150001, China
E-mail: chenbo@hit.edu.cn

A straightforward way to detect an unknown worm is to use various anomaly detection systems [1]. In the anomaly "intrusion detection" research area, there are many well-studied methods or systems such as IDES [2], NIDES [3] and EBayes [4]. Most of the anomaly intrusion detection systems concentrate on detecting attacks from hackers. Although the propagation of worms and the intrusion attack of hackers can both make servers and internet abnormal, there are still many differences between them. First, the worm's propagation action is simple, that is, all infected computers send out the similar packets continually. Moreover, the worm's propagation in the internet usually follows a dynamic model because of its large-scale distributed infection. Whereas, a hacker's intrusion attack, which is more complicated, usually aims at one or a set of targeted computers and does not follow any well-defined dynamic model in most cases. Therefore, this system is not suitable for worm detection.

For the reasons above, a new methodology is presented for defending against worm. Our algorithm first detects the occurring worm attacks utilizing computers' connection degree. The connection degree is defined to be the number of the hosts with different IP addresses that the given host wants to connect with in a time slot t. When a host was infected by the worm, it will send out a lot of connection requests. It will be shown as the quick augment of connection degree within the same time section. The worm attacks can be detected using this character. When a worm attack is found, the worm's packets are dropped to constrain its propagation. The rest of this paper is organized as follows. Section 2 surveys the related work and following that Section 3 briefly descries the defense mechanism. Section 4 presents a network monitoring algorithm and then an algorithm is presented for defending against worm in Section 5. In Section 6 the defending algorithm is analyzed and then in Section 7, some experiments are carried out to evaluate the algorithms. Finally the last section concludes this paper.

2 Related work

In the area of worm attack modeling, Kephart, White and Chess carried out a series of studies from 1991 to 1993

on viral infection based on epidemiology models [5–7]; Staniford et al. [8] adopted the classical epidemic model to model the spread of Code Red right after the Code Red incident in 2001; Ellis et al. [9] discussed the life cycle of worm and built a relational model. Zoue et al. [10] presented an improved worm propagation model, called two factors worm propagation model. In this model, the following two factors which affect worm propagation are considered.

First, human's countermeasures will result in removing both susceptible and infectious computers from circulation. That is, during the course of Code Red propagation, an increasing number of people became aware of the worm and would implement some countermeasures such as cleaning compromised computers, patching or upgrading susceptible computers, setting up filters to block the worm traffic on firewalls or edge routers, or even disconnecting their computers from Internet.

Second, define a degressive infection rate $\beta(t)$ but not a constant rate β anymore. The large-scale worm propagation has caused congestion and trouble to some Internet routers, thus it will slow down the Code Red scanning process.

In recent years, people have paid much attention to monitoring the Internet for malicious activities. Moore et al. [11] presented the concept of "network telescope" using a small fraction of unused IP space for observing security incidents in the global Internet. Berk et al. [12] brought forward a monitoring system by collecting ICMP "Destination Unreachable" messages generated by routers for packets to unused IP addresses. Staniford et al. [13] proposed an intrusion detection system called GrIDS, which can detect worm-infected hosts in a local network through building the worm's infection graph. GrIDS is an effective tool to defend Internet worm. However, it has a fatal disadvantage that GrIDS cannot detect unknown Internet worms. Zou et al. [1] presented a new methodology to detect worms by monitoring traffic trend. In this methodology, the system attempts to detect the dynamic trend of monitored traffic based on the fact that, at the early stage, worm propagates exponentially with a constant, positive exponential rate. Therefore, when a worm breaks out, the "trend" that is detected is the exponential growth trend of monitored traffic.

For a fast spreading worm such as Slammer, it is necessary to have automatic response and mitigation mechanisms. Moore et al. [14] discussed the effect of Internet quarantine for containing the propagation of a worm. Williamson [15] proposed a general rate-limiting "throttling" method to greatly constrain infection traffic sent out by infected hosts while not affect normal traffic. Zou et al. [16] presented worm propagation under feedback dynamic quarantine system for automatic mitigation by employing two principles utilized in the epidemic disease control in the real world "preemptive quarantine" and "feedback adjustment". In Worm Behavior Model and Propagation Model, Staniford [17] presented automatic worm quarantine for enterprise networks by using "Counter Malice" devices to separate an enterprise network into many isolated sub-networks.

3 Defense mechanism against worm attack

The framework of the defense mechanism is shown in Fig. 1. The whole system is composed of two parts: a data processing center (DPC) and distributed sensors. Our sensors are located on gateways or border routers of local networks. These sensors are deployed in monitoring network, detecting worms and computing worm attack status. Then these sensors will adaptively drop the packets which are sent by worm agents according to the worm attack status. The sensor's structure is shown in Fig. 2. Data processing center is responsible for receiving the result of each distributed sensor and computing the number of computer which is infected by the worm.

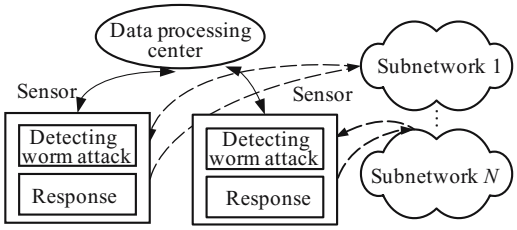

Fig. 1 A framework for defending against worm attack

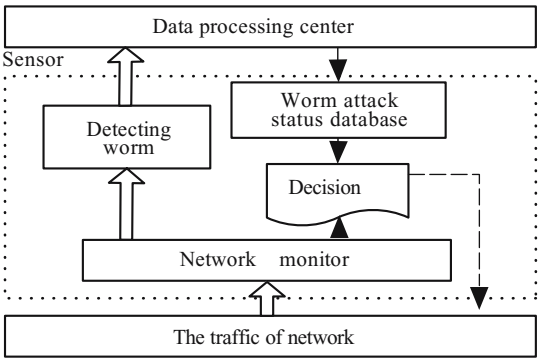

Fig. 2 Sensor's structure

4 Network monitoring algorithm

A network worm is defined as a process that can copy itself to remote computational machine and execute automatically [13]. It actively scans computers in network to detect if there are vulnerabilities. When it finds that a computer has vulnerability, the worm will send out the copy of itself to the remote computer. The general framework of network worms is

```
While(1)
{
    h = ScanAndGetTarget();
    acquiirePrivil (h,e);
    if can not acquire privilege then
        continue;
    infect (h);
    if can not infect then
        continue;
}
```

It can be found that the worm can be divided into the following three phases: 1) target acquisition phase; 2) elevating privilege phase; and 3) infection phase.

The target acquisition phase describes the process that a worm agent selects hosts for infection. In this phase, the worm agent will scan network to select a computer to infect. Therefore, it can be concluded that an infected host should have the following characteristics.

a) Within the same time section, the infected host has numerous connection requests. Figure 3 shows several kinds of typical worm's connection states in 30 s. Figure 4 is a normal host's connection states in 30 s in the morning, noon and evening respectively. Comparing Figs. 3 with 4, it can be found that within a same time section, the number of the infected host's connection request is far more than a normal host.

b) Destination ports are similar when the worm agent executes, because the sending rate of worm is far greater than the sending rate of normal behavior. Therefore, when a computer is infected by worms the destination port of packets which is sent by this computer is similar.

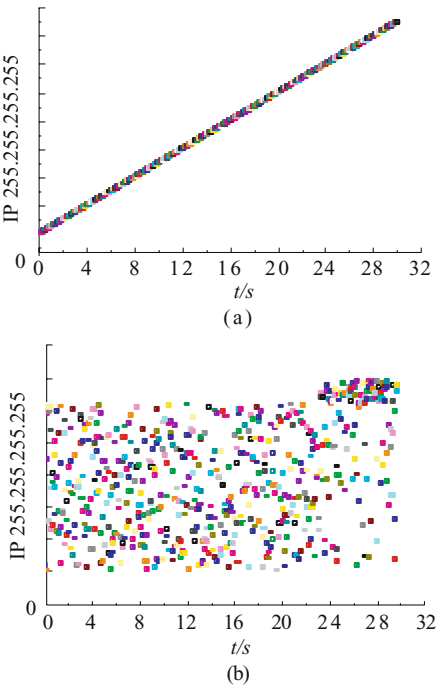

Fig. 3 Typical worms' connection states
(a) Connection state of Blaster; (b) connection state of Nimda

Definition 1 Connection degree: in a time slot t, host i contact n on different IP host computers. Then n is defined the connection degree of host i in time interval t.

From the analysis above, it can be known that the number of connection request of the host infected by worm is far greater than normal host. That is to say, the connection degree of the host infected by worm is far higher than normal host. Therefore, an algorithm can be designed to monitor computers' connection degree.

For computer i, it should have three states: 1) sending packets state; 2) receiving packets state; and 3) idle state.

Because of the character of a worm, little attention is paid to the receiving packets state. Therefore, binary time series $\{W(t),t\geqslant 0\}$ can be defined. $W(t)=1$ means that the computer is sending packets at time t and $W(t)=0$ means that the computer does not send packets at time t. Let $\{\boldsymbol{D}(t,t_1,t_2),t_1\leqslant t<t_2\}$ denote a series of different IP addresses, to which computer i is sending packets from time t_1 to time t_2. And let $D'(t)$ denote the destination computer's IP address to which computer i is sending packets at time t. Using the definition of connection degree mentioned above, the following equation can be obtained

$$C_t^i=\int_{(t-1)T}^{tT}W(t)K(t)\mathrm{d}t \tag{1}$$

where C_t^i is computer i's connection degree at time t. And $K(t)$ is a decision function. It can be expressed by

$$K(t)=D'(t)\oplus\boldsymbol{D}(t,0,t-1) \tag{2}$$

In Eq. (2), a new operator $\oplus$ is defined. It is an operation between a variable and a vector. Suppose that there is a variable a and a vector $\boldsymbol{A}$. It can be defined that

$$a\oplus\boldsymbol{A}=\begin{cases}0 & a\in\boldsymbol{A}\\ 1 & a\notin\boldsymbol{A}\end{cases} \tag{3}$$

Via utilizing Eq. (1), the algorithm can be got to record each computer's connection degree. The following shows the main steps of our monitoring algorithm.

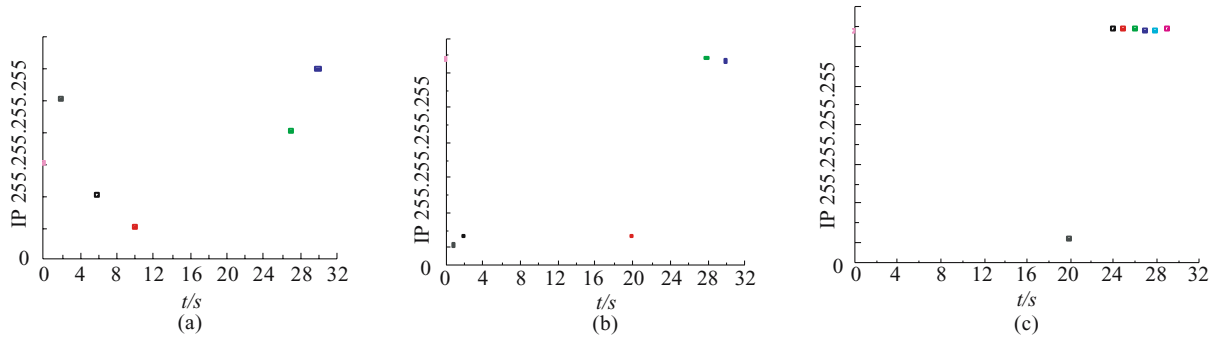

Fig. 4 Normal host's connection status
(a) Normal host's connection state at 10 a.m.; (b) normal host's connection state at 2 p.m.; (c) normal host's connection state at 8 p.m.

```
K SrcIP List;
L DstIP List;
P Destination Port List;
C packets sent by computer
   for each C(srcIP, srcPort, dstIP, dstIPort)
      if srcIP is in K
         if (dstIP is not find correspoind L)
            insert dstIP into correspoing L
            insert dstPort into P
            K[pos].count + = 1;
         else continue
      else
         insert srcIP into K
         insert dstIP into corresponding L
         pos = GetPosFormK (srcIPK)
         K[pos].count = 1
   end for
```

5 Algorithm for defending against worm attack

For a fast spreading worm, it is necessary to have automatic response and mitigation mechanisms. When it finds a host that is infected by worm, it can immediately isolate the host within seconds or milliseconds. In this way, the defense actions can catch up with a worm's fast infection speed and constrain the worm's propagation. In this section, an algorithm will be described to protect network from worm attack. Because any worm detection algorithm will generate false alarms and if the alarmed host is quarantined, then many healthy hosts will be quarantined for a long time. Such quarantine will dramatically interfere with normal activities. Therefore, in this method, it will drop suspicious traffic with probably ϑ. The following shows our algorithm.

```
Thread 1:
for each C_t^i
      find abnormal computers;
end for
compute worm attack status;
Thread 2:
for each packet (srcIP, srcPort, dstIP, dstPort)
      if (packet is attack packet) then
         get probability ϑ;
         drop the arriving packet with probability ϑ;
end for
```

The algorithm has two separate sub-algorithms. Algorithm 1 first detects abnormal computers and then computes that how many worms exist and how many computers are infected by each worm. In this paper, a data structure D which is a set of the form (p, I) is defined to save the worm attack status, where p is destination port which is used by worm to send packets, I is the number of infected host by this worm. Algorithm 2 first judges whether there are attack packets against network. In this paper, it is determined that whether the packet is attack packet on the basis of two conditions.

1) The computer which sends this packet belongs to abnormal computer.

2) The destination of this packet exists in D. If attack packets arrive, our algorithm first calculates ϑ for these packets, and then drops these packets with ϑ.

5.1 Anomaly detection algorithm

Our detection algorithm relies on testing whether the computer's connection degree exceeds a particular threshold. In order to account for seasonal variations and trends, the value of the threshold is set adaptively on the basis of an estimation of the mean number of computers' connection degree.

If x_n^i is the connection degree of computer i in the nth time interval, and $\bar{\mu}_n$ is the mean rate estimated from measurements prior to n, then it is predicted that the computer is infected by worm on the basis of the following condition

$$x_n^i \geqslant (1+\gamma)\bar{\mu}_{n-1} \tag{4}$$

where $\gamma > 0$ is a parameter that indicates the percentage above the mean value that is considered to be an indication of anomalous behavior. The mean $\bar{\mu}_n$ can be computed over some past time window or using an exponential weighted moving average (EWMA) of previous measurements

$$\bar{\mu}_n = \lambda\bar{\mu}_{n-1} + (1-\lambda)\frac{1}{G}\sum_{i=1}^{G} x_n^i \tag{5}$$

where λ is the EWMA factor. And G is the number of computers monitored.

Direct application of the above algorithm would yield high numbers of false detection. Therefore, the developed algorithm is used to improve its performance. The worm after a minimum number of consecutive violations of the threshold is detected.

$$\sum_{j=n-k+1}^{n} 1_{x_j^i \geqslant (1+\gamma)\bar{\mu}_{j-1}} \geqslant k \tag{6}$$

where $k > 1$ is a parameter that indicates the number of consecutive intervals and the threshold must be violated for an detection to be raised.

5.2 Computing worm attack status

The first task of computing worm attack status is to find a port which is probably used by worm to send packets. In this paper it is called suspicious port. And the second task of computing worm attack status is to get the number of computers which are infected by worm and how many worms exist. The general framework of computing worm attack status is shown as follows. It first monitors abnormal computers to find the suspicious ports, and then it updates the ***D*** based on monitoring result.

```
D initially is empty;
for each abnormal computer
    statistical sampling of incoming traffic to find the
    suspicious port vector P
    for each p in P
        if p exist in D
            increment frequency of I by 1;
        else
            add an item (p,1) into D;
    end for
end for
```

For our algorithm of computing worm attack status, its main task is to find suspicious ports. Suppose that a traffic of computer i can be described by port stream $S = \{s_1, s_2, \ldots\}$. Let p denote the destination port of packets, and then it is predicted that the port is suspicious port based on the following condition

$$f(p) > (1+\gamma)\overline{\mu}_n$$

where freq(x) obtains the frequency of item x. Then the task to find the suspicious ports can be described as: find items in stream s where $f(s) \geqslant \zeta N$ and $\zeta = (1+\gamma)\overline{\mu}_n / N$. In this paper, LCA [18] is used to solve this problem.

In LCA algorithm, it has a data structure S which is as set of entries of the form (e,f, Δ), where e is an element of stream, f is an integer representing its estimated frequency. And Δ is the maximum possible error in f. The LCA algorithm divides the incoming stream into buckets of width $w = 1/\xi$ transactions, where $\xi \in (0, 1)$ is an error parameter. Buckets are labeled with bucket id starting from 1. The LCA first records items' frequency and updates S in each bucket. And at bucket boundary, the algorithm will delete some item from S, in which $f + \Delta \leqslant b_{\text{current}-1}$. The LCA is showed as follows.

```
/*b_current denotes the current bucket id, the value of which
is ⌈N/w⌉, where N is current length of stream. */
S is initialize by empty;
for each element e
  if e exists in S
     increase its frequency by 1;
  else
     create a new entry (e, 1, b_current − 1);
  if is bucket boundory
     (e, f, Δ) is deleted when f + Δ ⩽ b_current − 1
end for
output those entries in D where f ⩾ (s − ε)N When a
request for getting items with threshold s,
```

5.3 Dropping packet mechanism

Let $\boldsymbol{P}$ (srcIP, srcPort, dstIP, dstPort) denote the network packet. If dstPort $\ni \boldsymbol{D}$, it is considered that the packed is sent by the worm and subsequently our defending algorithm will drop it. Accounting for the false alarm of our detection method, our algorithm must drop the packet adaptively. It will drop packet with probability ϑ. The more infected computers, the greater the probability ϑ. Their relation can be expressed as

$$\vartheta = \frac{1}{1 + A\mathrm{e}^{-Jt}} \tag{7}$$

where A, J are adjusting factors and A, J are adjusted to decrease or increase the changing speed of ϑ.

6 Defending algorithm analysis

Our worm detection algorithm will raise false alarms for healthy hosts from time to time. Suppose a healthy host exceeds $(1+\alpha)\overline{\mu}_i$ with probability p_1. Then the healthy host will be falsely alarmed by our detection algorithm with probability p_1^k. If let p_{nd} denote the probability that the normal packets are dropped, then p_{nd} can be expressed as

$$p_{\text{nd}} = p_1^k \frac{1}{1 + A\mathrm{e}^{-Jt}} \tag{8}$$

If $\gamma = 0$ and the connection degree follows the normal distribution, then it can be concluded that $p_1 = 0.5$. And then Eq. (8) can be substituted as

$$p_{\text{nd}} = 0.5^k \frac{1}{1 + A\mathrm{e}^{-Jt}} \tag{9}$$

Form Eq. (9), it can be known that if appropriate parameters are select, our algorithm basically does not interfere with healthy host.

Whether our algorithm is effective to constrain the worm propagation will be analyzed. When a worm breaks out in network, worm's propagation will follow fluid models due to its large-scale distributed infection. It is assumed that each host resides in one of the two states: susceptible or infected. When a host is infected by worm, it will remain in the infectious state forever. The model for a finite population is [14,16,19]

$$\frac{\mathrm{d}I_t}{\mathrm{d}t} = \beta I_t \left[N - I_t\right] \tag{10}$$

where I_t is the number of infected hosts at time t; N is the size of vulnerable population before any of them is infected; and β is called the pairwise rate of infection in epidemic studies. I_0 is the number of initially infected hosts.

Due to our defending against mechanism, at any time t the host will have three states: susceptible, infections and quarantined. If let $S(t)$ denote the number of susceptible hosts at time t, $I(t)$ denote the number of infectious hosts at time t and $Q(t)$ denote the number of quarantined infectious hosts at time t, the new interaction between $S(t)$ and $I(t)$ can be got as shown in Fig. 5.

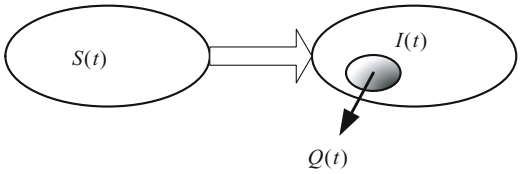

Fig. 5 Interactions between infectious and susceptible

Figure 5 shows that the interactions now are between $S(t)$ and $I(t)-Q(t)$. Therefore, a worm's propagation under the detection and quarantine algorithm follows

$$\frac{\mathrm{d}I(t)}{\mathrm{d}t}=\beta[I(t)-Q(t)][N-I(t)] \tag{11}$$

When a host which is infected by worm sends out a packet, our algorithm will interrupt this transmission at the probability of ϑ. Therefore, at any time t, quarantine $Q(t)$ can be expressed as

$$Q(t)=\frac{I(t)}{1+\mathrm{e}^{-\vartheta I(t)}} \tag{12}$$

Substitute Eq. (12) into Eq. (11), the following equation can be got

$$\frac{\mathrm{d}I(t)}{\mathrm{d}t}=\beta' I(t)[N-I(t)] \tag{13}$$

where β' can be expressed as

$$\beta'=\frac{A\beta}{A+\mathrm{e}^{\vartheta I(t)}} \tag{14}$$

Equation (14) shows that under our detection and quarantine algorithm, a worm still propagates but with slower spreading speed. Our algorithm can decrease a worm's pairwise rate of infection β by the adaptive factor of $A/(A+\mathrm{e}^{\vartheta I(t)})$, and the adaptive factor varies with $I(t)$. When $I(t)$ becomes large, β' will become small. Therefore, our algorithm that well constrains the worm propagation is implemented.

7 Experiments

In this section, extensive simulation experiments are carried out to evaluate this method. The structure of simulation platform is shown in Fig. 6. It is composed of four parts: the module of sending background noise, the module of sending worm's packet, controlling module and our adaptive worm detection and quarantine algorithm. The controlling module monitors the packet on network. If it finds a packet sent by module of sending worm's packet, it will start another module to send worm's packet with a possibility of μ, which is not constant. It will vary with the number of infected computer and it can be expressed by

$$\mu=\frac{N-I_t}{2^{32}} \tag{15}$$

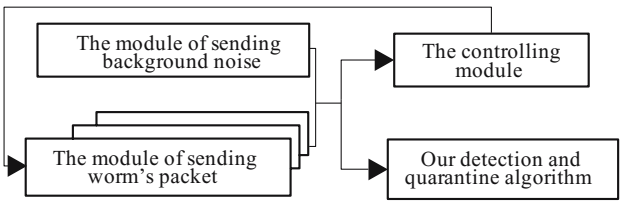

Fig. 6 Simulation platform of experiment

In this simulation experiment, background no-worm noise needs considering in the simulation. Goldsmith provided simple data of the background noise for Code Red activities. His monitored result showed that the background noise was small compared with Code Red traffic and the noise did not vary much. Zou et al. [1] proposed that the normal distribution model $N(20, 30)$ is appropriate for the number of scans and hosts that send noise. The parameter of our defending algorithm also needs considering. In this simulation, these parameters are $\gamma=0.98$, $k=3$, $\gamma=5$ and $A=10\,000$.

7.1 Single worm attack in network

Fist, a single worm attack is simulated. In this simulation experiments, the Code Red worm is simulated. In the case of Code Red, we set population $N=360\,000$. And assume that the scan rate of each infected hosts follows the normal distribution $N\left(\mu,\sigma^2\right)$, where $\mu=358$, $\sigma=80$. It also is assumed that $I_0=10$, i.e. ten vulnerable hosts in the system are infectious at the beginning.

Figure 7(a)–(c) shows the result of our monitoring algorithm within $k-$consecutive time and Fig. 7(d) shows

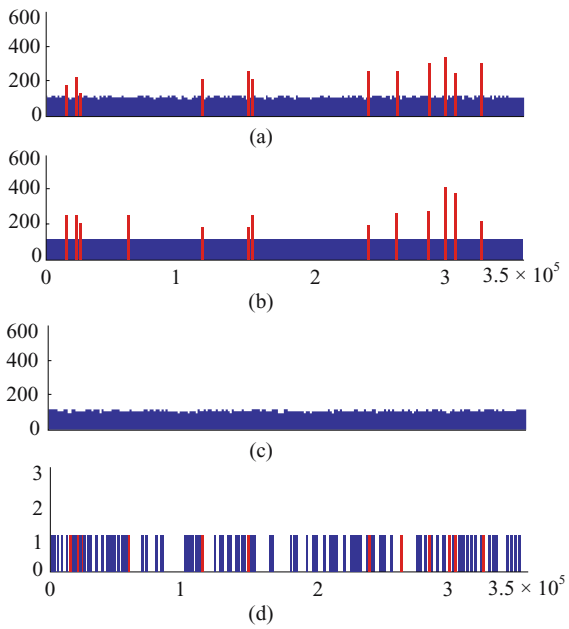

Fig. 7 Computers' connection degree within k - consecutive time and the detection result
(a) At $k-2$ time; (b) at $k-1$ time; (c) at k time; (d) detection result

the result of our detection algorithm. It can be known from Fig. 7(d) that our algorithm can detect which computer is abnormal but with false alarm.

When the abnormal computer is detected, our algorithm can find suspicious port and drop packets which are sent by worm. Figure 8 compares the number of infected hosts $I(t)$ and the number of infected hosts under our defending algorithm $I'(t)$. This figure shows that under our defending algorithm, the worm still propagates. However, the speed of propagation is very slow. And when the worm has infected 1 480 hosts, the worm almost cannot propagate. In Fig. 9, the number of dropped packets and transmitting packets is showed. Through Fig. 9, it can be known that our defending algorithm can effectively protect the network against worm attack. First when I_t is small, our algorithm only suspect that there is a worm attack in network, so it will drop packets with small probability. And with more and more hosts are infected by worm. Our algorithm adjusts dropping probability ϑ automatically according to $I(t)$. And it will verify that there is worm attack in network. Therefore, our algorithm will drop most packets which are sent by the worm to protect network.

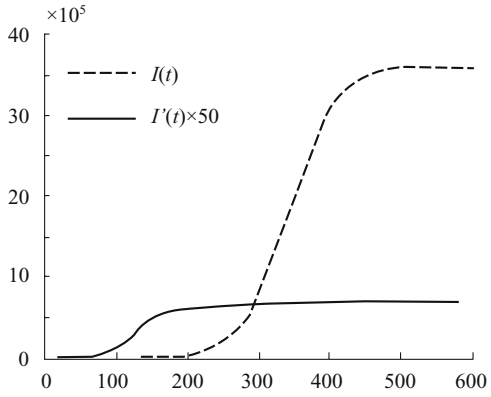

Fig. 8 Worm propagation comparison

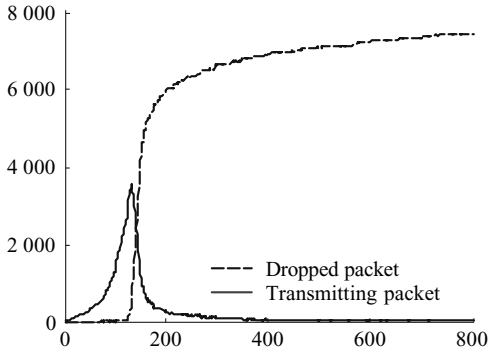

Fig. 9 Number of dropped packet S and the transmitting packets

A good defending algorithm not only constrains worm propagation well but also should not disturb normal behaviors of the host dramatically. Suppose N_T denotes the number of normal packets and B_N denotes the number of normal packets which is drop by our algorithm, then $B_N = p_{nd}N_T$. Figure 10 shows N_T and B_N when our quarantine algorithm is working. And Fig. 11 shows each normal host's average p_{nd} when our algorithm works. Through Fig. 11 it can be concluded that our defending algorithm almost does not interfere with healthy host activity when a single worm attacks the network.

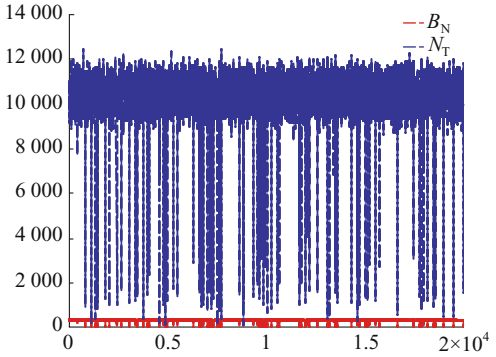

Fig. 10 Number of packet sent by normal host and number of normal packets dropped by our algorithm when only one single worm attack

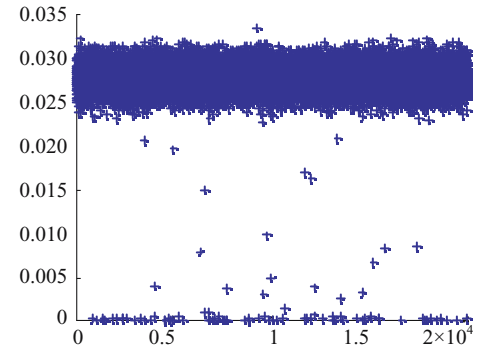

Fig. 11 Average p_{nd} of each host when only one single worm attack

7.2 Multiple worm attack in network

In this section, it is considered that there are two worm attacks in network. Table 1 describes the two worm's parameters. Figure 12 shows the number of worm attack packets and the number of packets dropped by our algorithm. Through these figures, it can be known that our algorithm can effectively defend against multiple worm attack. Figure 13 shows the number of normal packets which is dropped by our algorithm. And Fig. 14 is the average p_{nd} of each host when multiple worms attack. Through Fig. 14, it can be found that the average p_{nd} is very small. Our algorithm probably drops a packet when a normal host sends 1×10^5 packets. These

Table 1 Parameters of two worms

	Destination port	Scan rate	Vulnerable host	Initial infected hosts
Worm I	135	358	360 000	10
Worm II	4 444	214	240 000	2

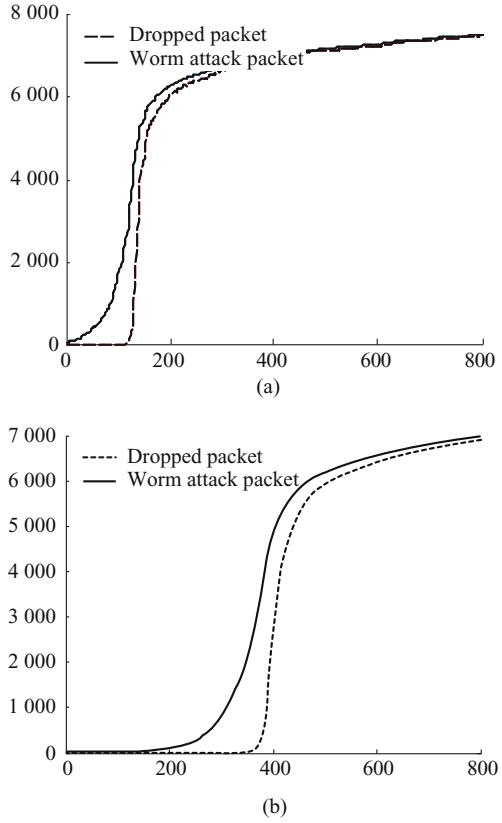

Fig. 12 Number of packet sent by worm attack and the number of packets dropped by our algorithm. (a) Worm I; (b) worm II

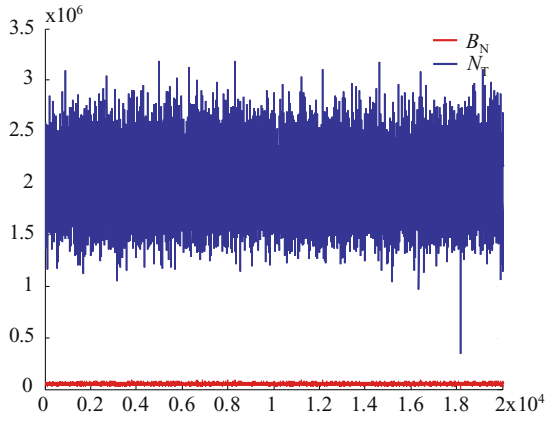

Fig. 13 Number of packet sent by normal host and normal packets dropped by our algorithm when multiple worms attack

results show that our algorithm can effectively work on multiple worm attack in network.

8 Conclusions

In this paper, a mechanism is described to defend against worm attack. Our mechanism includes the distinct features as follows.

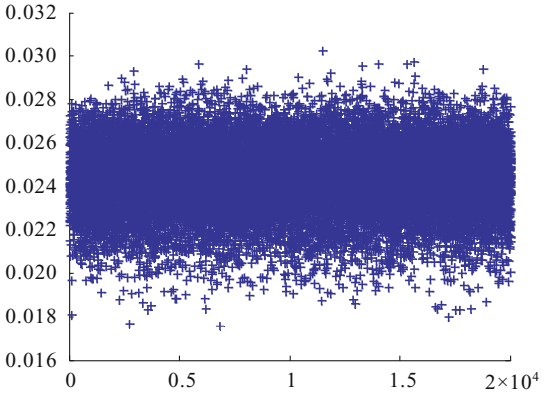

Fig. 14 Average p_{nd} of each host when multiple worms attack

1) Because whether a computer is abnormal is judged on the basis of its connection degree, the algorithm can significantly speed up the detection time.

2) In our algorithm, the worm attack status is saved, so even multiple worms exist in network, our algorithm will constrain these worms effectively.

3) When a packet is coming, it will be dropped with probability ϑ. This makes our algorithm possess small p_{nd} even in the worst case. At last, the extensive experiments have clearly confirmed that our defending algorithm can well constrain the worm propagation, in addition, it almost dose not interfere with the healthy host activity.

Acknowledgements This work was supported by the National Natural Science Foundation of China (No. 60403033).

References

1. Zou C C, Gong W B, Towsley D. The monitoring and early detection of internet worms. IEEE/ACM Transactions on Networking, 2005, 13(5): 961–974
2. Denning D. An intrusion detection model. IEEE Transactions on Software Engineering, 1987, 13(2): 222−232
3. Anderson D, Frivold T, Valdes A. Next-generation Intrusion Detection Expert System: A Summary. Technical Report SRI-CSL-95-07, SRI International, 1995
4. Valdes A, Skinner K. Adaptive, model-based monitoring for cyber attack detection. In: Proceedings of the 3rd International Symposium on Recent Advances in Intrusion Detection (RAID), Lecture Notes in Computer Science. Berlin: Springer-Verlag, 2000, 1 907: 80–92
5. Kephart J O, Chess D M, White S R. Computers and epidemiology. IEEE Spectrum, 1993, 30(5): 20–26
6. Kephart J O, White S R. Directed-graph epidemiological models of computer viruses. In: Proceedings of IEEE Symposium on Security and Privacy. IEEE Press, 1991, 343−359
7. Kephart J O, White S R. Measuring and modeling computer virus prevalence. In: Proceedings of IEEE Symposium on Security and Privacy. IEEE Press, 1993, 2–15
8. Staniford S, Paxson V, Weaver N. How to own the Internet in your spare time. In: Proceedings of USENIX Security Symposium. USA: USENIX, 2002
9. Ellis D. Worm anatomy and model. In: Proceedings of ACM workshop on Rapid Malcode. New York: ACM Press, 2003, 42–50

10. Zou C C, Gong W B, Towsley D. Code Red worm propagation modeling and analysis. In: Proceedings of the 9th ACM Conference on Computer and Communications Security. New York: ACM Press, 2002, 138–147
11. Moore D, Shannon C, Voelker G M, et al. Network Telescopes: Technical Report. Technical Report TR-2004-04, CAIDA
12. Berk V H, Gray R S, Bakos G. Using sensor networks and data fusion for early detection of active worms. In: Proceedings of the SPIE eroSense. USA: SPIE Press, 2003, 5 071: 92–104
13. Staniford C S, Cheung S, Crawford R, et al. GrIDS-A graph-based intrusion detection system for large networks. In: Proceedings of the 19th National Information Systems Security Conference, Baltimore, 1996, 361–370
14. Moore D, Shannon C, Voelker G M, et al. Internet quarantine: Requirements for containing self-propagating code. In: Proceedings of IEEE INFOCOM. IEEE Press, 2003, 1 901–1 910
15. Williamson M M. Throttling viruses: Restricting propagation to defeat mobile malicious code. In: Proceedings of the 18th Annual Computer Security Applications Conference. USA: IEEE Computer Society, 2002, 61–68
16. Zou C C, Gong W B, Towsley D. Worm propagation modeling and analysis under dynamic quarantine defense. ACM CCS Workshop on Rapid Malcode, Washington DC, USA, Oct. 27, 2003, 51−60
17. Weaver N, Staniford S, Paxson V. Very fast containment of scanning worms. In: Proceedings of the 13th USENIX Security Symposium, San Diego, CA, USA. USA: USENIX, 2004, 29–44
18. Manku G S, Motwani R. Approximate frequency counts over data streams. In: Proceedings of 28th International Conference on Very Large Data Bases. San Mateo: Morgan Kauffman Publishers, 2002, 346−357
19. Daley D J, Gani J. Epidemic Modeling: An Introduction. Cambridge: Cambridge University Press, 1999