A DoS/DDoS Attack Detection System Using Chi-Square Statistic Approach

Fang-Yie Leu*
Department of Computer Science, Tunghai University
leufy@thu.edu.tw
Contacting author

and

I-Long Lin
Dept. of Information Management, Central Police University, Taiwan
paul@mail.cpu.edu.tw

ABSTRACT

Nowadays, users can easily access and download network attack tools, which often provide friendly interfaces and easily operated features, from the Internet. Therefore, even a naive hacker can also launch a large scale DoS or DDoS attack to prevent a system, i.e., the victim, from providing Internet services. In this paper, we propose an agent based intrusion detection architecture, which is a distributed detection system, to detect DoS/DDoS attacks by invoking a statistic approach that compares source IP addresses' normal and current packet statistics to discriminate whether there is a DoS/DDoS attack. It first collects all resource IPs' packet statistics so as to create their normal packet distribution. Once some IPs' current packet distribution suddenly changes, very often it is an attack. Experimental results show that this approach can effectively detect DoS/DDoS attacks.

Keywords: Intrusion Detection System, DoS, DDoS, Mobile agent, Chi-square

1. INTRODUCTION

Denial of service (Dos) is a type of attack in which a hacker issues a huge amount of packets to congeal specific servers' services, consequently blocking legitimate users from normal access to the services. Distributed DoS (DDoS) attacks are another form of DoS attacks in which a host or hosts suffer from receiving a huge amount of packets issued by zombies. DDoS attacks often do not rely on particular network protocols or system weaknesses [1]. Instead, they simply exploit a tremendous amount of Internet resources, i.e., compromised hosts located between themselves and the victim (or victims), to send a huge amount of useless packets toward the victim (or victims) around the same time. The magnitude of the combined traffic is generally sufficiently huge to jam, or even crash, the victim (system resource exhaustion), or its Internet connections (bandwidth exhaustion), or both, consequently taking the victim off the Internet [1].

Often hackers of DoS attacks spoofed their attack packets' source addresses, and in DDoS attacks, each zombie only sends a limited amount of packets to a victim or victims. Both make it very difficult to trace to the real attackers [2]. According to a 2007 CSI Computer Crime and Security Survey [3], DoS attacks were in the top 5 among all attack types. 25 percent of respondents' computers had detected DoS attacks. Amount of companies' loss caused by DoS is $2,888,600 which was ranked the top 7 among all attack types. These show the severity of information security.

According to a 2008 CSI Computer Crime and Security Survey [4] about attacking technologies used, intrusion detection and intrusion prevention systems are very important tools for security. This survey also described that DoS attack has severely influenced on network security in a year after year tendency of the rise.

The surveys imply a fact that although present DoS is very severe to computer networks and systems, and many intrusion detection methods [5-11] have been developed, none of current detection approaches can completely protect a system and prevent a system from DoS attacks. The key reason is hackers discover new vulnerabilities and then create new attacking methods almost everyday. What we can do right now is continuing developing protection frameworks and algorithms to enhance security systems.

In this article, we propose a distributed security system, Agent-based Instruction Detection System (AIDS), to detect DoS and DDoS attacks. In these attacks, a huge number of connection-requests and/or data packets are sent to the same destination address, i.e., the victim. AIDS uses mobile agents to decentralize tasks of data analysis, and employs distributed components to reduce workload of detection tasks. A network management unit [12], e.g., a company's or a university's network system, can employ an AIDS as its security system to detect DoS/DDoS attacks.

The contributions of this research include (1) developing the AIDS, a distributed system with scalability; (2) Once AIDS has detected an attack, it sends hackers' information to a database, from which a firewall can accordingly drop known hackers' packets and disconnect hackers' connections. That is, this security system integrates firewall to mitigate the attack in a real-time manner; (3) AIDS can more effectively detect DoS/DDoS attacks than other tested approaches.

The rest of this article is organized as follows. Section 2 describes related work of this thesis. Section 3 introduces our system architecture and the algorithms that are used to detect DoS/DDoS attack. Experiments and discussion are stated in section 4. Section 5 concludes this article and addresses our future work.

2. BACKGROUND AND RELATED WORK

Feinstein et al. [13] used chi-square formula to detect DoS attacks. The authors classify network connections into six groups, called previous groups, according to packets' receiving frequencies, and calculate the total frequency of normal connections for each group. When detecting attacks, the detector again classifies frequencies of current connections, and classify then into six groups, called current groups. It then compares the frequency of previous group i and that of current group i with chi-square statistic method to see whether or not the difference is significant. If so, we can suspect that there is a DoS or DDoS attack.

However, several problems exist in this system. The first is that there are only six groups. For a huge institute or organization, the number should be increased. Otherwise, too many IPs are in a group, resulting higher false positives. The second, there is no way to identify who the backers are once a DoS/DDoS is discovered. To solve this problem, we create a mechanism to do this. The third, if hackers issue a DoS/DDoS attack with a specific protocol, e.g., icmp flooding, due to only occupying a portion of total packets, the attack is hard to be detected. The fourth is before calculating chi-square statistics, the authors grouped IPs based on their current connection frequencies rather than following the classification of previous groups. That is, a packet which is classified into previous group i may be classified into current group j, $1 \le i, j \le 6, i \ne j$, again resulting in the fact that we only know that there is a DoS/DDoS attack, but do not know who is issuing the attack. The final problem is the authors failed to considered network consumption attacks. They only considered resource consumption attack. In fact, network consumption attack can be detected by the same method.

2.1 Scenario of Distributed DoS attack

A DDoS attack often has two issuing stages to set up an attack, including control stage and attack stage. In control stage, the hacker looks for vulnerable systems and install handlers/masters and zombies/daemons by exploring system vulnerabilities. Famous DDoS master and agent programs include Trinoo, Tribe Flood Network 2000, and Stacheldraht.

In a DDoS attack, the handlers are the first level vulnerable hosts controlled by the attacker. The zombies are the second level vulnerable hosts controlled by the attacker through the handlers. Most of the control messages in control stage are single direction from attacker and handler, but is bi-direction between handlers and zombies. After the control stage, the list of vulnerable hosts is then entering their attack stage and launching a DDoS attack.

2.2 Mobile agents

A mobile agent is a program which has the ability to migrate among heterogeneous network systems. It can autonomously determine when to transfer to another system and where it can or should move to. An IDS that deploys mobile agents to detect attacks has advantages over a traditional IDS in that [14,15] it can
(1) Reduce network load: Mobile agents can execute programs stored in servers where they current reside to avoid issuing many requests and transferring a huge amount of data to and from other servers. This can reduce network burden and speed up data processing.
(2) Overcome network latency: Real-time systems do not permit long network latency. A mobile agent can monitor and control an underlying environment and respond appropriately and immediately.
(3) Encapsulate protocols: In a distribution system, although information is exchanged by using specific communication protocols, two heterogeneous systems with different communication protocols are hard to communicate with each other. However, a mobile agent can encapsulate packets of other protocols into a new packet as the new packet's payload, and send the new packet to next node. This is known as tunneling. Information can be then exchanged.
(4) Execute independently and autonomously: Mobile agents can autonomously perform tasks that users assign to them. They can also autonomously and independently determine when to migrate to other nodes and where to migrate.
(5) Adapt dynamically: A mobile agent can adjust its running method according to the change of its operational environment.
(6) Naturally heterogeneous: A network may consist of many heterogeneous nodes which are connected by network links, and which provide different operating systems and applications running on different hardware platforms. A mobile agent can adapt itself in such a heterogeneous environment. Also, when a host is going to be shut down, all mobile agents now residing on the host will be notified to migrate to other nodes so as to continue their operations and tasks.

2.3 Related Work

D-WARD [16] is a source-end DDoS defense system, which can identify malicious flows at the source end due to its architecture near to attack's source more effectively to discover attack, but it more difficult to deployment and detection. And its architecture contains statistics normal traffic models on three types including TCP, UDP and ICMP. Once an attack is discovered, it would control traffic, but D-WARD only deny by a rate limit. Actually, it will still impact on the performance of Internet. However, attackers can still successfully perform those attacks from unprotected network. In addition, those system require spending a huge amount money and time to deployed.

DefCOM [17] is a distributed framework for DDoS defense. It consists of heterogeneous defense nodes organized in a peer-to-peer network, communicating to achieve a dynamic cooperative defense and it carried out victim end, source end, and network core defenses mechanisms to perform attack detection, traffic differentiation and rate-limiting, respectively. Its components are of different types and can fulfill, e.g., only filtering or only attack detection. There are three types of nodes including alert generator to detect the attack and inform other nodes, classifier to distinguish legitimate traffic and rate-limiter. When under a DDoS attack, all nodes communicate with each other by flooding messages, however, this approach is only effective with all core router deployed at least. In addition, the compromised overlay nodes can do harm to the DefCOM operation. Another DDoS defense [18] allows implementing the cooperation of two and more perimeter defense systems.

The Distributed Intrusion Detection System (DIDS) [19] project was sponsored by the United States Air Force Cryptologic Support Center through a contract with the Lawrence Livermore National Labs. The DIDS architecture combines distributed monitoring and data reduction with centralized data analysis, but it did not scale well for large networks since addition of any new component increases the load on the DIDS director component, and the data flow from monitors to DIDS director consumes high network bandwidth. Our proposed approach focuses on scalability problems by using mobile agents to decentralize task of data analysis, and employs distributed components to reduce workload of detection tasks.

The Autonomous Agents for Intrusion Detection (AAFID) [20] made use of multiple layers of agents organized in a hierarchical structure with each layer performing a set of intrusion detection tasks. Administrator can send global instructions to all agents so that network and data can be respectively monitored and analyzed on each node. Data on end nodes are collected by agents which are dispatched by local monitoring nodes. Local monitoring nodes are responsible for analyzing data gathered by agents. Global monitoring nodes are in charge of integrity monitoring. However, AAFID uses only static agents and is deprived of some of the benefits mobile agents can offer.

The Lightweight agents for intrusion detection [21] had been developed for an IDS that deploys distributed multiple layers of lightweight intelligent mobile agents and applies data mining techniques to detect intrusions. An agent monitor system roams around different system networks, analyzing and integrating collected information and transferring the results to users and at last, storing the results into a database. This system allows an agent to increase its new ability during its execution period, and provides more convenient mechanisms to improve IDS's communication capability.

3. SYSTEM ARCHITECTURE

AIDS system architecture as shown in Figure 1 consists of five main components, including event monitoring subsystems, backup subsystems, mobile agents, a duty center and a black list database.

An event monitoring subsystem is employed to protect geographically concentrated subnets which are subnets geographically located together or nearby. A building with several subnets owned by different departments or the same department is an example. This subsystem monitors source and destination addresses of packets to detect whether there is a DoS/DDoS attacks. Once detected, it dispatches a mobile agent to send attacker's IP address to the black list database which is a database used to record hackers' information and intrusion details. In addition, it periodically dispatches another mobile agent to send the packet statistics to the duty center, which is the coordinator of an AIDS installed in a specific location, e.g., the computer center of a university, for further detecting whether or not there is a DDoS attack. If a DDoS truly exits, the duty center dispatches a mobile agent to record attackers' information also in the black list database. With the database, a firewall can accordingly filter out packets issued by known hackers.

When a host is under a DoS/DDoS attack and loses its detection capability, a node will be chosen from the corresponding backup subsystem and requested to substitute for the attacked one.

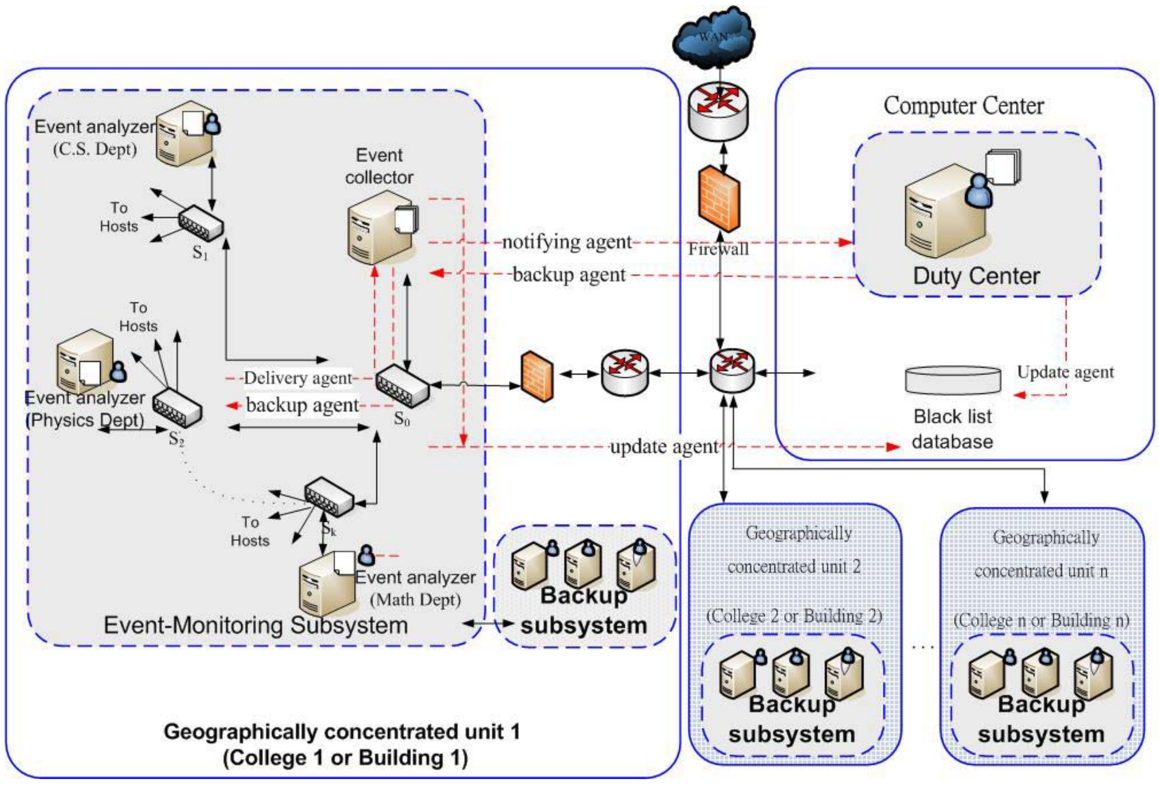

Figure 1. AIDS system architecture

Table 1. A source-IP distribution table (IP-based)

Source-IP-addr	Port #	Group #	Packet-information										10-sec-base	G%	N%	Chi-square %	Chi-square for amount
			Past 7th-day count	Past 6th-day count	Past 5th-day count	Past 4th-day count	Past 3rd-day count	Past 2nd-day count	Past 1st-day count	One-week count=$\sum_{i=1}^{7}$ ith day count	Current-day count	Past-10-sec count					

Table 2. A source-IP distribution table (group-based)

Group#	Packet-information					10-sec-base	G%	N%	Chi-square %	Chi-square for amount
	Past-7th-day-size	…	One-week size=$\sum_{i=1}^{7}$ ith-day size	Current day-size	Past-10-sec-size					

Event Monitoring Subsystem

An event monitoring subsystem as shown in Figure 1 consists of event analyzers and an event collector. An event analyzer is employed to detect DoS/DDoS attacks launched to the subnets that it protects. An event collector collects event information and packet statistics from its subordinate event analyzers, detects whether there is an DoS/DDoS attack and reports attacking information to the black list database.

An Event Analyzer: An event analyzer collects packets sent to hosts that it protects through a switch having a mirror port. A packet flowing through the switch will be duplicated and sent out via the mirror port from which an event analyzer gathers all its detection packets. The original packet will continue its journey to its destination. In addition, an event analyzer collects source and destination addresses of packets sent to protected subnets, and counts number of packets that each sender (source IP address) sends to a specific host or subnet in the protected subnets in order to produce a source-IP distribution table (which will be described later) from which the event analyzer can detect DoS/DDoS attacks, and identify who is issuing the attacks. An event analyzer periodically once per ten second dispatches a mobile agent, called delivery agent, to deliver the source-IP distribution table to its coordinating event collector. An event collector on receiving the table accumulates the table contents to it source-IP accumulation table which is a table used to accumulate packet statistics for the subnets in the underlying geographically concentrated unit.

(1) Source-IP distribution table: A source-IP distribution table as shown in Table 1 is an IP-based table. Each source IP has its own tuple to record the IP's packet information. This table consists of Source-IP, port#, group#, packet-information, 10-sec-base, group-percentage (G%), N-percentage (N%), chi-square-percentage, and chi-square for amount where Source-IP records source IP of a received packet, e.g., packet P, port# lists the ports that has been used by P's sender, group# identifies which group P belongs to. Details of groups will be described later. Packet-information consists of two portions, packet-information-amount and packet-information-size. The former (the later) is used to count number of packets (to accumulate size of packets) that P's sender has sent during a specific period of time. Packet-information-amount has 10 subfields, including past 7 days' counts, (i.e., past-ith-day-count, i=1,2,…7), one-week-count, current-day-count and past-10-sec-count. Past-ith-day-count records the past ith day's packet count, one-week-count= $\sum_{i=1}^{7}$ past-ith-day-count, current-day-count records packet counts for current day, and past-10-sec-count keeps information of past 10 seconds. Packet-information-size has also 10 subfields where field names are exactly the same as those of packet-information-count, except the "count" in each field name is substituted by "size" since these subfields are used to accumulate sizes of receiving packets rather than counts of the receiving packets. For example, one-week-size== $\sum_{i=1}^{7}$ past-ith-day-size which accumulates sizes of receiving packets for the past 7 days. However, to avoid redundantly listing them, Table 1 only lists packet-information-count portion. 10-sec-base records the 10 sec average of an IP's normal traffic in the past one week. Group-percentage represents the percentage of packet counts that a group received previously under normal circumstance over the total number of packets that the system has received, N-percentage is defined as the number of packets that a group has currently received, e.g., *Qi*, over total number of packets the network has received, i.e., *Q*, in the past 10 seconds (i.e., N-percentage= Q_i / Q), chi-square-percentage defined as $(N\% - G\%)^2 / G\%$ is used to analyze whether or not there is a resource consumption attack, chi-square for amount defined as $\frac{((past - 10 - \sec - size) - (10 - \sec - base))^2}{10 - \sec - base}$ is employed to analyze whether or not there is a bandwidth consumption attack. However, in the following, we remove field name of packet-information to make all its subfields to be level-one fields. Further, to clearly illustrate behaviors of a group instead of individual behaviors of single IPs, we create another table, called group-IP distribution table, which calculates values for different group features. The values are obtained by summing up field values recorded in the corresponding source-IP distribution table.

The group-IP distribution table, as shown in Table 2, has the same attributes as those of Table 1. But omitting Sorrce-IP-address and port# since in a group-based table, the two fields lose their original features.

From the group-IP distribution table, we can realize which group or groups, e.g., groups1', 2',…*k*', k'≦13, are issuing DoS/DDoS attacks. Therefore, the search space of finding out who are launching the attacks can be reduced to the IPs that belong to groups 1', 2', …k'. Now, we can realize who is issuing the attack by checking packet-information-amount and packet-information-size subfields and source-IP field in their original source-IP distribution table. The source-IP can be used to trace back to the hacker. In addition, we can close the ports that P's sender has used once the sender is suspected as a hacker.

(2) Baseline Profile Establishment: Before detection, we first collect ordinary packets for subunits of a geographically concentrated unit for one week, filter out attack packets and calculate their normal packet counts and normal accumulated packet sizes. The IP with the largest amount of packets is ranked number one. The second largest amount is ranked number two, and so on. After that, we classify them into 13 groups as follows. That ranked number one constitutes the first group, i.e., group 0. Those ranked number two and three are group 1. The classification is shown in Figure 2.

The baseline profile is updated every 24 hours. Each time after update, we recalculate chi-square value $\overline{x_i}$ for each group i, i=0,1,2,……12, as their new baseline profile values, where $\overline{x_k} \geq \overline{x_j}$ when *k>j,* k=0,1,2,…………11, and j=1,2,3,……12.

Group#	Range of source IPs	Ranked no.
Group 0	2^0	1
Group 1	2^1~2^2-1	2,3
Group 2	2^2~2^3-1	4,5,6,7
Group 3	2^3~2^4-1	8,9,10,11,12,13,14,15
Group 4	2^4~2^5-1	16,17,18,……………31
Group 5	2^5~2^6-1	32,33,34,……………63
Group 6	2^6~2^7-1	64,65,66,……………127
Group 7	2^7~2^8-1	128,129,130,………..255
Group 8	2^8~2^9-1	256,257,258,………..511
Group 9	2^9~2^{10}-1	512,513,514,……….1023
Group 10	2^{10}~2^{11}-1	1024,1025,1026,…...2047
Group 11	2^{11}~2^{12}-1	2048,2049,…………4095
Group 12	2^{12} and up	4096 and remaining

Figure 2 classification of collected source IPs

(3) Algorithms: Algorithms for establishing a source-IP distribution table and Baseline Profile are as follows:

```
Algorithm 1: Establishing a source-IP distribution table $T_{D-IP}$
Input: An incoming packet P with source IP S-IP and destination IP D-IP;
Output: update S-IP's information in $T_{D-IP}$, or insert a new tuple with source IP address=S-IP into $T_{D-IP}$
{1. If (S-IP is already in $T_{D-IP}$, e.g., tuple t) { /* i.e., t.source-IP = S-IP */
  1.1 t.past-10-sec-count ++;
  1.2 t.past-10sec-size =t.past-10sec-size +$|P|$; /*$|P|$: packet size*/
  1.3 If (P's port# is not in "port#" field) Append the port# to "port#" field; /* accumulating port # */}
 Else { /* S-IP is absent from $T_{D-IP}$ */
  1.4 Retrieve port # from P; , e.g., port j;
  1.5 Insert(source-IP=S-IP, port #=port j, past-10-sec-count=1, past-10sec-size=$|P|$;) into $T_{D-IP}$; /* insert a new tuple, and other fields are given null values*/ }
 2. If (timer times out) {
     /*accumulate current-day-count/size, reset past-10-sec count/size and detect DoS/DDoS per 10 seconds*/
  2.1 t.current-day-count=t.current-day-count + t.past-10-sec-count;
  2.3 t.current-day-size=t.current-day- size + t.past-10-sec-size;
  2.4 t.past-10-sec-count=0;
  2.5 t.past-10-sec-size=0;
  2.6 Call Algorithm 3; /*detecting DoS/DDoS attack per 10 seconds*/
  2.7 Set timer to 10 seconds;}
```

The algorithm that establishes a baseline profile is as follows.

```
Algorithm 2: Establishing a baseline profile /* summing up the past 7-day's counts/sizes and classifying source IPs in a source-IP distribution table into 13 groups */
Input: a source-IP distribution table T which contains two tables, T-count and T-size
Output: tuples in T are classified into 13 groups
{If (timer times out) /* timer's initial value=24 hr */
  1.1 For ( i=6; i<=1; i--)
      {Shift past-i th-day-count to past-(i+1) th-day-count;
       Shift past-i th-day-size to past-(i+1) th-day-size;}
  1.2 Shift current-day-count to past-1 th-day-count;
  1.3 Shift current-day-size to past-1 th-day-size;
  1.4 For (each tuple in T )
      {t.one-week-count=$\sum_{i=1}^{7}$ t. past-i th-day-count;
       t.one-week-size=$\sum_{i=1}^{7}$ t. past-i th-day-size;}
  1.5 Call Sort-data (current-day-count, T-count=T); /* generating baseline profile for count */
  1.6 Call Sort-data (current-day-size, T-size=T); /* generating baseline profile for size */}
  1.7 For (each tuple q in T-count and T-size)
      Fill in the group number to which q belongs to group# field;
  1.8 Set timer t=24 hr;}

Sort-data (x, y) { /* x may be current-day-count or current-day-size */
  1.1 sort tuples in T on x subfield;
  1.2 The IP ranked top one is x-group 0;
  1.3 The IPs ranked the top 2nd and top 3rd are x-group 1;
  1.4 The IPs ranked the top 4th to top 7th are x-group 2;
  1.5 The IPs ranked the top 8th to top 15th are x-group 3;
   …
  1.6 The IPs ranked the top $(2^k)$ th to top $(2^{k+1}-1)$th are x-group k;
   …
  1.7 The IPs ranked the top $(2^{11})$ th to top $(2^{12}-1)$th are x-group 11;
  1.8 The IPs ranked the top $2^{12}$ and up are x-group 12;}
```

(4) Detecting DoS/DDoS attacks: The method to detect a DoS/DDoS attack is as follows. Let $G_{i\text{-count}}$ be average number of packet counts that group I, i.e., N_i, has received per 10 seconds under the circumstance of no attacks, $G_{i-count} = \frac{\sum_{t \in Ni} t.one-week-\text{count}}{\frac{7*24*60*60}{10}}$. Let $G_{i\text{-size}}$ be average accumulated packet sizes that group i has received per 10 seconds also under the circumstance of no attacks, $G_{i-size} = \frac{\sum_{t \in Ni} t.one-week-\text{size}}{\frac{7*24*60*60}{10}}$. The threshold values calculated beforehand for each group are as follows. Let

$x^2_{count} = \sum_{i=0}^{n-1} \frac{N_{i-count} - G_{i-count}}{G_{i-count}}$, and $x^2_{count} = \sum_{i=0}^{n-1} \frac{N_{i-count} - G_{i-count}}{G_{i-count}}$

where $N_{i-count}(N_{i-size})$ is number of packets (accumulated packet size) that group i has currently received in past 10 seconds, i=0,1,2,…12, n=13, and degree of freedom is df=12 (=n -1).

In this research, we choose significant standard $\alpha = 0.05$ and $x^2_{12,\alpha} = 21.026$. If $x^2 \le x^2_{12,\alpha}$, we accept the null hypothesis H_0, indicating that there is no attack where x^2 may be either x^2_{count} or x^2_{size}. However, if $x^2_{count} > x^2_{12,\alpha}$ (or $x^2_{size} > x^2_{12,\alpha}$), we reject H_0, which means its alternative hypothesis H_1 is true, showing that there is a suspected resource consumption (bandwidth consumption) attack.

Algorithm 3: detecting DoS attack by using chi-square statistic method invoked by an event analyzer for every ten seconds.

Input: a source-IP distribution table T; baseline profile-count; baseline profile-size

Output: whether or not a subnet or subnets protected by an event analyzer are under a DoS/DDoS attack

```
{1. Att=false;
 2. If $(x^2_{count} \ge (x^2_{12,\alpha} + threshold))$ /* a resource consumption attack is discovered */
    {For (i=0; i<=12; i++) /* check which groups issued the attack */
      If ( $\frac{(N_{i-count} - G_{i-count})^2}{G_{i-count}} > threshold_{i-count}$ )
        Choose k IPs whose past-10-sec-counts are the highest as the suspected hackers, where
        $\sum_{j=0}^{j=k-1} past-10-\sec-count_j / \sum_{j=0}^{m_i-1} past-10-\sec-count_j$ in $\ge 80\%$
        which $m_i$ is number of source IPs in group i;
      Send attack information to the black list database, and an
          alter message to administrator to show that there is a
          resource consumption DoS/DDoS attack;}
 3. If $(x^2_{size} \ge (x^2_{12,\alpha} + threshold))$ /* a bandwidth consumption attack is discovered */
    {For (i=0; i<=12; i++) /* check which groups issued the attack */
      If ( $\frac{(N_{i-size} - G_{i-size})^2}{G_{i-size}} > threshold_{i-size}$ )
        Choose k IPs whose past-10-sec-sizes are the highest as the suspected hackers, where
        $\sum_{j=0}^{j=k-1} past-10-\sec-size_j / \sum_{j=0}^{m_i-1} past-10-\sec-size_j$ ; $\ge 80\%$
     Send attack information to the black list database, and an
         alter message to administrator to show that there is a
         resource consumption DoS/DDoS attack;}}
```

An Event Collector

As stated above, an event analyzer periodically, once per 10 seconds, sends its source-IP distribution table to its event collector which on receiving the table, for each source IP address retrieves the corresponding count and size from the table, sums up the counts and sizes for each packet information subfield (a total of 10+10 subfields) and records the results in the corresponding subfields in its own source-IP accumulation table, which has the same schema as that of Table 1. In other words, the values of packet information subfields of a source IP address P appearing in different source-IP distribution tables will be summed up as total counts/sizes of the corresponding subfields of P in a source-IP accumulation table. For example, in the source-IP accumulation table, P's past-ith-day-count is the sum of P's past-ith-day-counts in all the received source-IP distribution tables, $1 \le i \le 7$.

An event collector should receive source-IP distribution tables periodically from all its event analyzers. When it does not receive the table from an event analyzer, it dispatches a mobile agent as a backup agent to check status of the event analyzer. If the event analyzer is still alive, the backup agent asks the event analyzer to send the source-IP distribution table to the event collector. Otherwise, the backup agent will request the corresponding backup subsystem to select a host to take over for the failed event analyzer. The functions of the backup host are exactly the same as those of an event analyzer.

An event collector detects whether a geographically concentrated unit is attacked by a DoS/DDoS or not also based on chi-square statistic approach. It calculates chi-square values of normal network traffic in advance for a geographically concentrated unit by using the algorithm similar to Algorithm 3 with the given source-IP distribution table substituted by its source-IP accumulation table. Its baseline profile values are also generated from the accumulation table beforehand. Each time, when an event collector finishes collecting source-IP distribution tables and updating its own source-IP accumulation table, of course once per 24 hours, it periodically once per 10 seconds checks to see whether the chi-square value of current network traffic for its 13 groups significantly exceeds its threshold or not. If yes, it dispatches a mobile agent as a update agent to send hacker information to the black list database. And regardless of yes or not, it dispatches a mobile agent as a notifying agent to send its source-IP accumulation table, also per 10 seconds, to the duty center, which will integrate contents of source-IP accumulation tables delivered by all event collectors, into its source-IP integration table from which the duty center can further detect DDoS attacks, particularly for low-density DoS/DDoS attacks, which are hard to be detected by an event collector.

Duty Center

The duty center, as the coordinator of AIDS, further detects whether or not there is a DoS/DDoS attack which is attacking the protected system. The duty center builds a source-IP integration table, of which the table structure is similar to that of a source-IP distribution table, to detect attacks. If yes, the duty center dispatches a mobile agent to record attackers' information also in the black list database. With the database, a firewall can accordingly filter out packets issued by known hackers. Algorithm 4 shows the detection details of the duty center.

Algorithm 4: process of the duty center

Input: source-IP accumulation tables received periodically

Output: whether network management unit, e.g., a university/company, is under a DDoS attack

```
{1. If (timer generated for an event collector times out)
     {Dispatches a mobile agent as a backup agent to check status
         of event collector;
      If (event collector fails)
          Send a message to the corresponding backup subsystem
          to choose a host to take over for the failed event collector;
```

```
        Else asks the event collector to send content of its source-IP accumulation table to the duty center;}
  2. Integrate all source-IP accumulation tables received to generate its source-IP integration table;
  3. Detect whether there is a DoS/DDoS attack by calculating $G_{i-count}$, $G_{i-siez}$, $x^2_{count}$ and $x^2_{size}$ at duty-center level;
  4. If (yes)
     {Send a message to administrators;
      Dispatch an update agent to deliver hacking information to the black list database;}}
```

Backup Subsystem

When an event analyzer (an event collector) is under a Dos/DDos attack and loses its detection capability, as stated above, its event collector (the duty center) dispatches a backup agent to the corresponding backup subsystem, asking the subsystem to choose a host to substitute for the attacked event analyzer (an event collector). The process of selecting a backup host by an event collector is as follows.

```
Algorithm 5: process of choose a backup host for an attacked event analyzer
Input: a set of hosts H= {h_1,h_2,h_3,…h_m} in the underlying geographically concentrated unit, where m is number of hosts in the backup subsystem
Output: the chosen host acts as an event analyzer
{1. Choose the host with the highest performance, e.g., h_i;
 2. Send a message to request h_i acting as an event analyzer to take over for the failed event analyzer;
 3. Change h_i's network interface (i.e., network card) into promiscuous mode to filter packets sent to the protected subnet or subnets;}
```

The algorithm that the duty center invokes to select a backup host is similar to algorithm 5.

Mobile Agents

There are four types of mobile agents. The first is delivery agents which are those dispatched by event analyzers to send source-IP distribution tables to their event collector. The second is notifying agents which are those dispatched by event collectors to send their source-IP accumulation tables to the duty center. The third is update agents which are dispatched by event analyzers, event collectors or the duty center to send hacker information to the black list database. The fourth is backup agents which are dispatched by event collectors (or the duty center) to check an attacked event analyzer's (an attacked event collector's) status and/or select a backup node to substitute for the attacked one.

Black List Database

Black list database keeps hackers' information, e.g., hackers' source IP addresses, hacking date/time, victim IPs, protocols of attacking packets, etc. Using BLD has several advantages: (1) Identical attack information can be combined to form a record. (2) Intrusion warning can be significantly reduced. (3) If an event analyzer has detected an attack, we can prevent the attacker from sending attacking packets to other geographically concentrated units in advance. (4) Router or firewall can follow the black list in black list database to discard packets issued by hackers.

Security of AIDS

Readers may ask how AIDS protects itself from being attacked. All event analyzers, event collects and backup hosts are given private IPs. So they are not visible to hackers. The duty center is the only one that is given public IPs. Its security is implemented by using firewall and IDS.

4. EXPERIMENTS AND DISCUSSIONS

Our experimental environment comprises computers installed in computer classrooms of Science and Technology Building, Computer center and Engineering-college Building in Tunghai University. The configuration is shown in Figure 3, in which the only victim is located in Science and Technology Building, and number of attackers and their positions are listed in Table 3. Wireshark (version 0.99.5) software is installed on the victim to gather traffic issued by the attackers, and attack tools are used to send attack packets, which have their own features so that we can discriminate attack packets from normal ones.

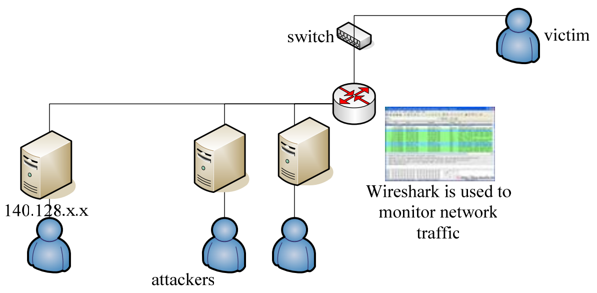

Figure 3. The configuration of our experimental environment

Table 3. Numbers of attackers and their positions

Building	Comp. room	No. of compu.
Sci. and Tech.	Room 1	20
Computer Center	Room 1	36
	Room 2	41
Eng.-college	Room 1	39

Table 4. The Information about Attacks

experiment	Attack Type	Attack Period	ANP/sec
1	Resource Consumption Attack	10 sec	323,201
2	Bandwidth Consumption Attack	10 sec	32,404

Table 5. Rations of attack packets distributed to each group in experiments 1 and 2.

Group	Experiment 1 (%)	Experiment 2 (%)
0	18.75	0
1	34.38	0
2	21.88	0
3	25	0
4	0	0
5	0	0
6	0	0
7	0	49.52
8	0	8.02
9	0	13.06
10	0	10.69
11	0	10.69
12	0	8.02
	100%	100%

Three experiments were performed in this study. Experiments 1 is a resource consumption attack, experiments 2 is a bandwidth consumption attack and experiment 3 evaluates detection accuracies of tested security systems.

Table 4 lists the attack details where ANP/sec stands for average number of launched packets per second. Each attack is separately issued twenty times. Table 5 lists how many percentages of attack packets are delivered by each group in the first two experiments. For example, in experiment 1, 18.75% of attack packets were sent by the hosts belonging to baseline-profile group 0, and 34.38% were delivered by members of baseline profile group 1 and so on.

(1) Experiments 1 and 2

In experiment 1, only 85.6%(=2,766,800/(323,201*10)) of packets arrived at the victim, including legitimate packets and attack packets. Table 6 lists the contents of a source-IP accumulation table established in this experiment. Table 7 lists the source-IP accumulation table generated in experiment 2. However, in this table, tuples Qs that belong to a group are summed up to be a tuple R to simplify scope of the table, where R's field i's value= $\sum_{j=1}^{|Qs|}($ tuple j's field i's value), i=1,2,3…k, in which k is number of fields that R and Qs have. We call the simplified table a concise accumulation table, from which, we can realize which groups are issuing DoS/DDoS attack.

Besides, packets were also further classified according to their protocols so that we can detect DoS/DDoS attacks of a specific protocol. Tables 8, 9 and 10 respectively list the baseline profiles for icmp, tcp and udp, which are obtained by classifying Table 7's tuples based on the tuples' protocols. Of course, the corresponding baseline sub-profiles for the protocols are also generated from the original baseline profile beforehand. So, when we are interested in detecting a specific type of DoS/DDoS, e.g., ICMP flooding, only ICMP packets are compared with ICMP baseline profile.

Table 6 Packet statistics collected in experiment 1 for an event collector's source-IP accumulation table and the chi-square result values.

Group	Source IP	Past 7th-day count	Past 6th-day count	Past 5th-day count	Past 4th-day count	Past 3rd-day count	Past 2nd-day count	Past 1st-day count	one-week count= $\sum_{i=1}^{7}$ ith- day count	current-day count	Past 10-sec count	10-sec-base	G%	N%	Chi-square %	Chi-square for amount
G0	140.128.0.132	6,634,149	7,016,363	10,579,147	11,070,566	10,374,389	11,466,430	11,111,517	68,252,562	7,553,961	537,598	1,128.51	15.83	19.43	0.82	475.38
G1	140.128.1.85	3,284,640	4,091,457	5,481,159	5,136,475	6,686,429	5,501,435	6,169,027	39,800,171	4,804,121	712,664	1,241.64	17.41	31.37	11.19	698.07
	140.128.0.125	4,014,561	3,628,274	6,699,195	6,277,914	5,929,474	6,723,976	5,470,646	35,294,492	3,783,604	155,330					
G2	163.23.75.125	1,161,892	1,228,833	1,816,951	2,008,209	1,946,050	1,852,812	1,938,878	11,953,623	1,688,232	459,399	790.58	11.09	22.21	11.16	776.41
	140.128.0.123	1,394,271	1,474,599	2,180,341	2,409,851	2,335,260	2,223,374	2,326,653	14,344,348	1,526,847	52,248					
	140.128.0.121	1,115,417	1,179,679	1,744,273	1,927,880	1,868,208	1,778,699	1,861,323	11,475,478	1,231,414	51,735					
	140.128.0.119	975,989	1,032,219	1,526,239	1,686,895	1,634,682	1,556,362	1,628,657	10,041,044	1,083,442	51,223					
G3	140.128.201.1	438,732	464,008	758,302	734,831	699,624	732,122	686,083	4,513,701	1,001,435	537,427	621.93	8.72	26.64	36.82	1,184.34
	140.128.0.74	548,415	580,011	947,877	918,538	874,530	915,153	857,603	5,642,126	613,572	33,561					
	140.128.0.110	475,293	502,676	821,494	796,066	757,926	793,132	743,256	4,889,843	535,198	32,522					
	140.128.0.109	402,171	425,341	695,110	673,595	641,322	671,112	628,909	4,137,559	457,743	32,402					
	140.128.0.2	365,610	386,674	631,918	612,359	583,020	610,102	571,736	3,761,418	417,638	30,964					
	140.128.0.124	292,488	309,339	505,534	489,887	466,416	488,082	457,388	3,009,134	339,704	30,365					
	140.128.0.100	621,537	657,345	1,074,261	1,041,010	991,133	1,037,173	971,950	6,394,410	677,881	20,536					
	140.128.0.99	511,854	541,343	884,685	857,302	816,228	854,143	800,430	5,265,985	560,760	19,417					
G4	140.128.0.98	168,992	178,728	283,044	269,483	282,001	264,267	292,085	1,738,600	179,041	313	459.95	6.45	0.18	6.09	5001
	140.128.0.97	189,213	180,632	217,001	255,023	312,012	255,763	364,102	1,773,746	180,945	313					
	…															
G5	140.128.0.95	93,860	99,268	149,674	156,626	146,777	162,227	157,206	965,638	99,349	81	510.92	7.17	0.09	6.98	2605
	140.128.0.93	94,025	100,234	153,247	179,621	158,140	186,203	201,001	1,072,471	100,316	82					
	…															
G6	140.128.0.91	76,119	80,504	127,021	119,033	131,563	127,491	121,383	783,113	80,514	10	828.69	11.62	0.02	11.58	612
	140.128.0.78	76,197	87,045	132,015	120,031	148,014	198,171	125,003	886,476	87,055	10					
	…															
G7	140.128.0.104	35,542	37,589	55,580	61,430	59,529	56,677	59,309	365,655	37,593	4	773.87	10.85	0.02	10.82	498
	140.128.0.14	35,961	38,763	56,237	62,301	60,521	57,023	60,201	371,007	38,767	4					
	…															
G8	140.128.0.24	20,155	21,317	17,418	16,879	16,070	16,817	15,759	124,417	21,318	1	438.86	6.16	0.01	6.13	316
	140.128.0.103	21,111	21,974	18,024	17,693	19,012	17,028	16,800	131,642	21,976	2					
	…															
G9	140.128.0.88	1,539	1,627	2,577	2,453	2,567	2,406	2,659	15,828	1,628	1	134.00	1.88	0.01	1.86	302
	140.128.0.66	1,589	1,852	2,896	2,566	2,596	2,963	2,912	17,374	1,853	1					
	…															
G10	140.128.0.66	508	537	847	794	877	850	809	5,222	537	0	88.42	1.24	0.00	1.24	68
	140.128.0.78	611	612	917	812	896	876	819	5,543	612	0					
	…															
G11	140.128.0.69	260	275	406	449	435	414	434	2,674	275	0	90.54	1.27	0.00	1.27	-1
	140.128.0.77	289	385	409	498	503	513	498	3,095	385	0					
	…															
G12	140.128.0.41	260	76	414	434	406	449	435	2,475	76	0	21.93	0.31	0.00	0.31	-1
	140.128.0.31	262	78	417	441	413	478	466	2,555	78	0					
	…															
Sum											2,766,800	7,129.84	100	100	106.26	

Table 7. Packet statistics collected in experiment 2 for an event-collector's source-IP accumulation table (the statistics of a group on each field are summed up as a value so the statistics of a group are represented by a tuple) and the chi-square values.

group	Past-7th-day-size	…	one-week size= $\sum_{i=1}^{7}$ ith-day size	current-day-size	Past-10-sec -size	10-sec -base	G%	N%	Chi-square %	Chi-square for amount
G0	775,727,731	…	7,545,989,602	933,126,289	403	124,768	9.66	0.00	9.66	1.00
G1	1,190,306,386	…	11,578,855,894	1,338,557,993	378	191,449	14.82	0.00	14.82	1.00
G2	977,342,839	…	9,507,226,058	1,489,806,973	757	157,196	12.16	0.00	12.16	1.00
G3	963,328,802	…	9,370,902,748	1,629,936,147	1,513	154,942	11.99	0.00	11.98	0.99
G4	1,092,567,924	…	10,628,092,647	1,752,168,605	3,027	175,729	13.60	0.01	13.59	0.98
G5	1,042,673,876	…	10,142,741,984	2,290,023,921	5,484	167,704	12.98	0.01	12.96	0.97
G6	694,202,739	…	6,752,944,934	1,016,441,048	428	111,656	8.64	0.00	8.64	1.00
G7	652,538,289	…	6,347,648,721	1,215,619,256	24,155,732	104,955	8.12	49.51	210.93	229.15
G8	469,953,146	…	4,571,528,658	644,275,288	3,912,800	75,587	5.85	8.02	0.80	50.77
G9	90,838,914	…	883,647,024	85,890,491	6,371,047	14,611	1.13	13.06	125.89	435.06
G10	41,083,945	…	399,649,273	38,845,909	5,212,675	6,608	0.51	10.68	202.92	787.85
G11	35,461,674	…	344,957,920	33,529,910	5,217,066	5,704	0.44	10.69	238.87	913.69
G12	8,179,848	…	79,570,504	7,734,253	3,912,800	1,316	0.10	8.02	627.11	2,972.25
sum			78,153,755,968		48,794,110		100	100	1,490	5,381.84

Table 8. ICMP packet statistics collected in experiment 2 (Table 7) for an event-collector's source-IP accumulation table

group	Past-7th-day-size	…	one-week size= $\sum_{i=1}^{7}$ ith-day size	current-day-size	Past-10-sec -size	10-sec -base	G%	N%	Chi-square %	Chi-square for amount
G0	5,376,553	…	55,314,334	11,288,487	24,155,732	914.59	3.36	49.52	634.45	26,410.57
G1	5,052,214	…	51,977,507	10,864,524	3,912,800	859.42	3.16	8.02	7.50	4,551.86
G2	10,104,427	…	103,955,013	23,119,680	6,371,047	1,718.83	6.31	13.06	7.22	3,705.61
G3	20,208,855	…	207,910,027	42,121,136	5,212,675	3,437.67	12.62	10.69	0.30	1,515.34
G4	40,417,709	…	415,820,053	77,050,021	5,217,066	6,875.33	25.25	10.69	8.39	757.81
G5	73,225,911	…	753,352,997	139,605,969	2,137,200	12,456.23	45.74	4.38	37.40	140.52
G6	4,582,857	…	47,148,732	13,522,551	1,762,800	779.58	2.86	3.61	0.20	2,740.49
G7	1,128,951	…	11,614,727	7,933,432	12,800	192.04	0.71	0.03	0.65	65.65
sum			1,647,093,390		48,782,120	27,233.69	100	100	696.1	39,887.86

Table 9. TCP packet statistics collected in experiment 2 (Table 7) for an event-collector's source-IP accumulation table

group	Past-7th-day-size	…	one-week size= $\sum_{i=1}^{7}$ i day size	current-day-size	Past-10-sec-size	10-sec -base	G%	N%	Chi-square %	Chi-square for amount
0	733,470,189	…	7,545,989,602	1,267,727,034	1,491	124,768.35	17.29	22.61	1.64	121,804.13
1	592,522,265	…	6,095,908,074	1,024,113,313	1,416	100,792.13	13.97	21.48	4.04	97,979.04
2	624,405,787	…	6,423,927,851	1,079,221,393	1,138	106,215.74	14.72	17.25	0.43	103,952.82
3	664,447,439	…	6,835,879,003	1,148,430,699	699	113,027.10	15.66	10.60	1.64	111,633.39
4	697,472,387	…	7,175,641,846	1,205,513,314	507	118,644.87	16.44	7.69	4.66	117,632.81
5	409,245,418	…	4,210,343,809	707,338,188	557	69,615.47	9.65	8.44	0.15	68,506.77
6	241,072,455	…	2,480,169,294	438,840,485	787	41,008.09	5.68	11.93	6.86	39,449.74
7	173,795,110	…	1,788,015,533	313,429,275	0	29,563.75	4.10	0	4.10	29,563.75
8	28,278,362	…	290,929,655	54,477,746	0	4,810.34	0.67	0	0.67	4,810.34
9	22,647,119	…	232,995,056	42,732,464	0	3,852.43	0.53	0	0.53	3,852.43
10	22,657,515	…	233,102,005	41,554,000	0	3,854.20	0.53	0	0.53	3,854.20
11	31,934,980	…	328,549,178	57,179,951	0	5,432.36	0.75	0	0.75	5,432.36
sum			43,641,450,907		6,595	721,584.84	100	100	26.01	708,471.78

Table 10. UDP packet statistics collected in experiment 2 (Table 7) for an event-collector's source-IP accumulation table

group	Past-7th-day-size	…	one-week size= $\sum_{i=1}^{7}$ ith-day size	current-day size	Past-10-sec-size	10-sec -base	G%	N%	Chi-square %	Chi-square for amount
0	657,074,690	…	6,760,027,671	1,135,685,430	1,220	111,772.94	18.74	22.61	0.80	109,346.41
1	684,774,023	…	7,045,000,233	1,183,560,796	1,159	116,484.79	19.53	21.48	0.19	114,178.42
2	401,055,323	…	4,126,083,565	693,183,552	931	68,222.28	11.44	17.25	2.95	66,373.53
3	456,110,937	…	4,692,499,353	788,342,918	572	77,587.62	13.01	10.60	0.45	76,447.99
4	344,921,694	…	3,548,577,103	596,166,007	415	58,673.56	9.84	7.69	0.47	57,846.67
5	316,336,665	…	3,254,492,435	546,755,159	455	53,811.05	9.02	8.44	0.04	52,904.15
6	311,015,208	…	3,199,744,934	539,552,828	644	52,905.84	8.87	11.93	1.05	51,626.30
7	255,199,682	…	2,625,511,130	441,086,079	0	43,411.23	7.28	0	7.28	43,411.23
8	62,417,969	…	642,160,170	107,883,128	0	10,617.73	1.78	0	1.78	10,617.73
9	14,333,837	…	147,467,459	30,376,097	0	2,438.28	0.41	0	0.41	2,438.28
10	2,884,232	…	29,673,167	46,169,969	0	490.63	0.08	0	0.08	490.63
sum			36,071,237,220		5,396	596,415.96	100	100	15.5	585,681.34

In Table 6 (experiment 1) Chi-square % field of the whole table is 106.26 which is very larger than 21.026, showing that there is an attack. Also, from this table, we can realize that groups 0, 1, 2 and 3 are issuing DoS/DDoS attack since their $\frac{\{(past-10-sec-count)-(10-sec-base)\}^2}{10-sec-base}$ hugely exceed their thresholds 1128.51, 1241.64, 790.58 and 621.93, respectively. Further, we can discover that 140.128.0.132, 140.128.1.85, 163.23.75.125 and 140.128.201.1 issued the attack since their past-10-sec-counts are the highest and are apart of the top 80% of packets.

When a tuple's 10-sec-base > Past-10-sec-count field or G% > N%, and the corresponding frequency value is below a certain threshold, even the corresponding chi-square % or chi-square for amount exceeds the threshold, the tuple can be purged without substantially affecting the chi-square computation.

In Table 7 (experiment 2), Chi-square for amount field of the whole table is 5,381.84 which is larger than 21.026, illustrating there is an attack. In this experiment, we would like to detect specific protocol attack, so we can check Table 8 if icmp attack is our concern. When detecting attacks, each cluster is compared with its corresponding protocol baseline sub-profile. Thus, we can discover a specific flooding attack, particularly when number of the flooding attack packets is not significantly increased compared with total packets currently collected in their source-IP accumulation table. However, when attackers mix TCP, ICMP and UDP flooding attacks, and each of them is not significant as compared with its own baseline sub-profile, this method can not discover such an attack. But the mixed attack can be discovered by comparing the mixed baseline profile.

(2) Experiment 3

In experiment 3, we evaluate detection accuracies of different tested security systems, including Kaspersky Anti-Hacker 1.8.180 (Kaspersky for short, by Kaspersky Labs), McAfee VirusScan Home Edition 7.0 (McAfee VirusScan forshort, by McAfee, Inc), Panda Titanium Antivirus 2005 (Panda Titanium for short, by Panda software), Snort and AIDS. We gather 100 times of normal traffic and 100 times of attack traffic of 10 seconds, respectively, including DoS/DDoS resource consumption attack (i.e., TCP flood and ICMP flood) and DoS/DDoS bandwidth consumption attack (i.e., UDP flood). The attack intensities range between 500 and 15,000 packets/sec [22].

Table 11 shows the detection results. Kaspersky performs the best (94.7%). But, if the detective host has been installed P2P software, its false positive is then high up to 6.8 %.

When Kaspersky and Snort discover that there is an attack, they throw the packets directly, and terminate the corresponding sessions. Panda Titanium has a specific phenomenon. During the experiment, it does not alert that there is an attack for many times of attacks. The error (false-positives) for CIDS comes from when G% and N% values are similar, consequently being treated as normal traffic.

Table 11 Detection Accuracy of DoS/DDoS attacks

Statistics / Secu. Systems	True Positive	False Negative	False Positive	True Negative	Detection Accuracy
Kaspersky	89.4%	100%	0%	10.6%	94.7%
McAfee VirusScan	89.5%	100%	0%	10.5%	84.75%
Panda Titanium	87.7%	100%	0%	12.3%	83.85%
Snort	89.5%	95.2%	4.8%	10.5%	82.35%
AIDS	92.76%	96.5%	3.5%	7.24%	94.63%

5. CONCLUSIONS AND FUTURE RESEARCH

In this paper, we proposed a distributed detection architecture called agent based intrusion detection system (AIDS), which uses Goodness of fit test of chi-square test to detect DoS/DDoS attacks. It analyzes amount and variation of source address that send packets to us, and statistics of IP address distribution. If hackers employ attack tools, for example, "Stacheldraht" to generate a huge amount of packets of random source IP addresses. We check to see whether its chi-square value exceed threshold. Experimental results show that this method can effectively detect DoS/DDoS attacks.

In the future, we would like to study AIDS's system behavior and reliability and develop the behavior and reliability models, so users can accordingly predict AIDS's system behavior and reliability before using it.

6. REFERENCES

[1] R.K.C. Chang, "Defending against Flooding-Based Distributed denial-of-Service Attack: A Tutorial," *IEEE Communication Magazine*, October 2002, pp. 42-51.

[2] W.Y. Luo , "A Lightweight System of Detecting DoS/Probe Attacks Based on Packet Header", National Taiwan University of Science and Technology, Dept. of Computer Science and Information Engineering, Master thesis,2008.

[3] "2007 CSI Computer Crime and Security Survey," http://i.cmpnet.com/v2.gocsi.com/pdf/CSISurvey2007.pdf , June. 2007. (visit on 2009/6/24)

[4] "2008 CSI Computer Crime and Security Survey," http://i.zdnet.com/blogs/csisurvey2008.pdf, June 2008. (visit on 2009/6/24)

[5] Guang Jin et al., "A Pi2HC mechanism against DDoS attacks," *Communications and Networking in China, 2008*, Auguster 2008, pp. 225-229.

[6] Zhou, Zaihong et al., "A P2P-Based Distributed Detection Scheme against DDoS Attack," *Education Technology and Computer Science,* March. 2009, pp. 304-309.

[7] I.B. Mopari et al., "Detection and defense against DDoS attack with IP spoofing," *Computing, Communication and Networking*, December 2008, pp. 1-5.

[8] R.P Karrer et al., "Joint Application and Network Defense against DDoS Flooding Attacks in the Future Internet," *Future Generation Communication and Networking,* December 2008, pp. 11-16.

[9] A. Chonka et al., "Multi-Core Defense System (MSDS) for Protecting Computer Infrastructure against DDoS Attacks," Parallel and Distributed Computing, Applications and Technologies, 2008, December 2008, pp. 503-508.

[10] Shen Wang et al., "GA-Based Filtering Algorithm to Defend against DDoS Attack in High Speed Network," *Natural Computation*, 2008, October 2008, pp. 601-607.

[11] R. Saad et al., "A collaborative peer-to-peer architecture to defend against DDoS attacks," *Local Computer Networks*, 2008, October 2008, pp. 427-434.

[12] F.Y. Leu, J.C. Lin and M.C. Li, "A Performance-Based Grid Intrusion detection system," *Proc. the International Computer Software and Applications Conference*, 2005, pp.525-530.

[13] L. Feinstein et al., "Statistical Approaches to DDoS Attack Detection and Response," *Proc. DARPA Information Survivability Conf. and Exposition*, 2003, pp. 303–314.

[14] W. A. Jansen, "Intrusion Detection with Mobile Agents"**,** *National Institute of Standards and Technology*, Apri1 2001.

[15] W. Jansen and T. Karygiannis, "Mobile Agent Security", *Computer Security Division*, NIST Special Publication 800-19, June 2001.

[16] J. Mirkovic and P. Reiher, "D-WARD: a source-end defense against flooding Denial-of-Service attacks," *IEEE Transactions Dependable and Secure Computing*, vol. 2, no.3, July 2005, pp. 216-232.

[17] J. Mirkovic, M. Robinson, P. Reiher, and G.. Oikonomou, "A framework for collaborative DDoS defense," *Proc. of the 22nd Annual Computer Security Applications Conference*, 2006, pp. 33-42.

[18] S. Chen, and Q. Song, "Perimeter-Based Defense against High Bandwidth DDoS Attacks", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 16, No. 7, 2005, pp. 526-537.

[19] S. Snap et al., "DIDS (Distributed Intrusion Detection System) – Motivation, Architecture, and an Early Prototype," *Proc. of National Computer Security Conference*, 1991, pp. 167-176.

[20] J. Balasubbramaniyan, et al. "An Architecture for Intrusion detection using Autonomous Agents," *Technical report No. TR 98-05*, Prudue University, 1009..

[21] G. Helmer et al. "Lightweight Agents for Intrusion detection," *Journal of systems and Software*, Elsevier, Vol. 67, 2003, pp. 109-122.

[22] R. Oliver, "Countering SYN Flood Denial-of-Service Attacks," *Tech Mavens, Inc.*, August 2001. http://www.tech-mavens.com/synflood.htm