Detecting DoS and DDoS Attacks using Chi-Square

Conference Paper · January 2009
DOI: 10.1109/IAS.2009.292 · Source: DBLP

2 authors, including:

Fang-Yie Leu
Tunghai University
227 PUBLICATIONS **681** CITATIONS

SEE PROFILE

Detecting DoS and DDoS Attacks using Chi-Square

Fang-Yie Leu and Chia-Chi Pai
Department of Computer Science, Tunghai University, Taiwan
leufy@thu.edu.tw g942922@thu.edu.tw

Abstract

In this paper, we propose an agent_based distributed intrusion detection architecture, which detects DoS/DDoS attacks by comparing source IP addresses' normal and current connection frequencies. First, we collect source IPs' packet statistics to obtain their normal packet distribution. When current statistics suddenly increase, very often it is an attack. Experimental results show that this approach can effectively detect DoS/DDoS attacks.

1. Introduction

Nowadays, users can easily access and download network attack tools from the Internet, which often provide friendly interfaces. Therefore, even a naïve user can also launch a large scale DoS or DDoS attack. 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. DDoS attacks are another form of DoS in which a host or hosts suffer from receiving a huge amount of packets issued by Zombies [1].

In this paper, we propose an agent_based distributed security system, called Agent-based Instruction Detection System (AIDS), to detect DoS and DDoS attacks. AIDS uses mobile agents to decentralize tasks of data analysis, and employs distributed components to reduce workload of detection tasks.

2. Related Work

The Distributed Intrusion Detection System architecture [2] combines distributed monitoring and data reduction with centralized data analysis. But it did not scale well for large networks. The Autonomous Agents for Intrusion Detection (AAFID) [3] made use of multiple layers of agents organized in a hierarchical structure with each layer performing a set of intrusion detection tasks. However, AAFID uses only static agents and is deprived of some of the benefits mobile agents can offer.

3. System Architecture

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

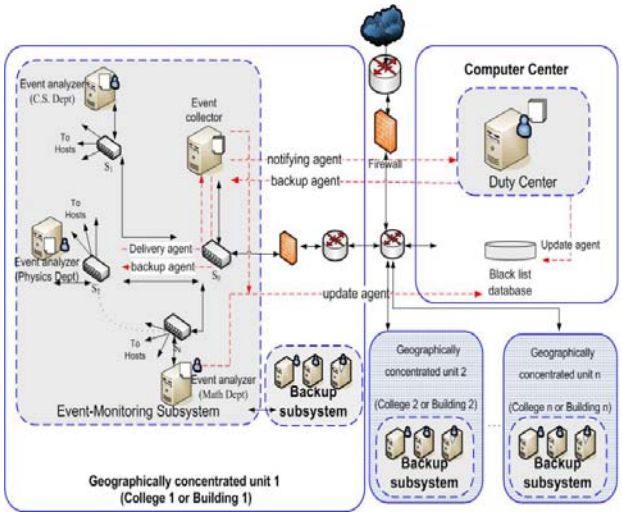

Figure 1. AIDS system architecture

3.1. Event Monitoring Subsystems

An event monitoring subsystem as shown in Figure 1 comprises event analyzers and an event collector, which are employed to protect geographically concentrated subnets. A building with several subnets owned by different departments or the same department is an example.

An event analyzer collects source and destination addresses for packets sent to hosts that it protects through a switch having a mirror port, and counts number of packets each sender (source IP address) sends to a specific host or subnet. A packet *P* flowing through the switch will be duplicated. The duplicated packet is then sent out via the mirror port from which an event analyzer gathers all its detecting packets. *P* will continue its journey to destination. In addition, an event analyzer creates a source-IP distribution table, which as shown in

978-0-7695-3744-3/09 $25.00 © 2009 IEEE
DOI 10.1109/IAS.2009.292

Table 1 has four main attributes, source-IP, port #, Group # and packet-information (packet-inf for short), where packet-inf is used to count number of and accumulates size of packets that *P*'s sender has sent during a specific period of time, and with which we can detect DoS/DDoS attacks, and identify who is issuing the attacks. It periodically, once per ten second, dispatches a mobile agent to deliver this table to its event collector to accumulate packets sent to the subnets in underlying building.

Table 1 A source-IP distribution table

Sour. -IP	Port #	Group-No.	Packet-inf					
			past 7^{th}-day-count /size	…	past 1^{st}-day-count /size	one-week sum up= $\sum_{i=1}^{7} i^{th}-day-count$	current-day-count /size	Past 10-sec-count /size

Before detection, we first collect ordinary packets for subunits of a geographically concentrated unit for one week to establish a baseline profile which is updated daily. After each 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. 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 which will be described in Algorithm 2. Algorithm 1 illustrates how a source-IP distribution table is established. Let $|P|$ be *P*'s packet size.

Algorithm 1: Establishing a source-IP distribution table $T_{D\text{-}IP}$

Input: An incoming packet *P* with source IP, S-IP, and destination IP, D-IP

Output: update S-IP's information in $T_{D\text{-}IP}$

{If (S-IP is already in $T_{D\text{-}IP}$, e.g., tuple t)
/* i.e., t.source-IP = S-IP */
{t.packet-inf.current-day-count++; t.packet-inf.current-day-size=t.packet-inf.current-day-size+$|P|$);
If (P's port# is not in "port#" field) Append the port# to "port#" field; /* accumulating port # */}
Else /* S-IP is absent from $T_{D\text{-}IP}$ */
{Retrieve port # from P; , e.g., port j;
Insert(source-IP=S-IP, port #=port j, group no.=null, packet-inf.current-day-count=1, packet-inf.current-day-size=$|P|$, packet-inf.past-10-sec-count=0, packet-inf.past-10sec-size=0;) into $T_{D\text{-}IP}$; /* a new tuple*/ }
If (timer times out) /*reset past-10-sec-count*/
{t.packet-inf.past-10-sec-count=0;
t.packet-inf.past-10-sec-size=0;
Set timer to 10 seconds;}}

Algorithm 2: Establishing a baseline profile-for count
/* summing up present 7-day's counts and sizes, and classifying source IPs */

Input: a source-IP distribution table T

Output: tuples in T are classified into 13 groups

{If (timer times out) /* timer's initial value=24 hr */
{For (i=1; i<=6; i++)
Shift i^{th}-day-count to $(i+1)^{th}$-day-count;
Shift current-day-count to 1^{th}-day-count;
For (each tuple t in T)
t.one-week sum-up-count=$\sum_{i=1}^{7}$ t.i^{th}-day-count;
Sort tuples in T on "one-week sum-up-count" field;
The IP ranked top one is group 0;
The IPs ranked top 2^{nd} and 3^{rd} are group 1;
The IPs ranked top 4^{th} to 7^{th} are group 2;
…
The IPs ranked top $(2^{11})^{th}$ to $(2^{12}-1)^{th}$ are group 11;
The IPs ranked top 2^{12} and up are group 12;
For (each tuple q in T)
Fill in "q.group-No" field the group number to which q belongs}
Set timer=24 hr;}

In algorithm 2, a baseline profile is established for count. The baseline profile for size is also established by the similar algorithm with count being replaced by size. To detect DoS/DDos attacks, let

$$G_{j-count} = \frac{\sum_{t \in Nj} t.one-week \text{ sum-up.count}}{\frac{7*24*60*60}{10}} \quad \text{and} \quad G_{j-size} = \frac{\sum_{t \in Nj} t.one-week \text{ sum-up.size}}{\frac{7*24*60*60}{10}}$$

be respectively average numbers of packet counts and sizes that group i, denoted by N_j, have received per 10 seconds when there is no attacks. Let

$$x^2_{count} = \sum_{0}^{n-1} \frac{(N_{i-count} - G_{i-count})^2}{G_{i-count}} \quad \text{and} \quad x^2_{size} = \sum_{0}^{n-1} \frac{(N_{i-size} - G_{i-size})^2}{G_{i-size}}$$ where

$N_{i\text{-}count}$ ($N_{i\text{-}size}$) is number of packets (accumulated packet size) that group i has currently received in past 10 seconds, n=13, and df=(n-1=12) is degree of freedom. In this research, we choose significant standard $\alpha = 0.05$, so $x^2_{12,0.05} = 21.026$.

If $x^2 \leq x^2_{12,\alpha}$, 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. The algorithm invoked to detect resource consumption DoS/DDoS attacks is shown in algorithm 3.

Algorithm 3: detecting resource consumption DoS/DDoS attacks with chi-square method by an event analyzer

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

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

{1. Att=false;

2. If (timer times out) /* initial value is 10 sec*/
If $(x^2_{count} \geq (x^2_{12,\alpha,count} + threshold))$
{For (i=0; i<=12; i++)
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 hackers, where $\sum_{j=0}^{j=k-1} 10-\text{sec}-count_j / \sum_{j=0}^{m_i-1} 10-\text{sec}-count_j \geq 80\%$, in which m_i is number of source IPs in group i;
Send an alert message to duty center and administrator to show that there is a resource consumption DoS/DDoS attack;
Send source IP addresses that issue the attacks to the black list database;}}

The algorithm for detecting bandwidth consumption attacks is similar to algorithm 3 with count being replaced by size.

An event collector on receiving source-IP distribution tables from all its event analyzers once per 10 seconds accumulates the table contents to its source-IP accumulation table, of which the table structure is similar to that of a source-IP distribution table, and checks to see whether the chi-square values of current network traffic (including size and count) for its 13 groups significantly exceed event-collector-level thresholds or not. If yes, there is an attack which is attacking subnets protected by some of its event analyzers with low-density traffic. And regardless of yes or no, it dispatches a mobile agent to send its source-IP accumulation table to the duty center.

3.2. 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: detection process of the duty center
Input: source-IP accumulation tables received periodically
Output: whether a network management unit, e.g., a university/company, is under a DoS/DDoS attack
{1. If (timer of an event collector E times out)
{Dispatches a mobile agent as a backup agent to check status of E;
If (E fails)
{The agent chooses the host with the highest performance, e.g., h_j, from the corresponding backup system, and asks the host acting as an event collector to take over for E;
Change h_j's network interface (i.e., network card) into promiscuous mode to filter packets originally sent to E;}
Else asks E 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 $G_{i\text{-count}}$, $G_{i\text{-siez}}$, 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;}}

3.3. Backup Subsystem

When an event analyzer is under a Dos/DDoS attack and loses its detection capability, its event collector dispatches a backup agent to the corresponding backup subsystem to choose a host and requests the host acting as an event analyzer to substitute for the attacked one. The process of selecting a backup host by an event collector is as follows.

Algorithm 5: choosing a backup host for an attacked event analyzer
Input: a set of m hosts H= $\{h_1,h_2,h_3,\ldots h_m\}$ in the underlying geographically concentrated unit.
Output: the chosen host acts as an event analyzer
{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 as an event collector is described in algorithm 4.

4. Experiments and Discussion

Our experimental environment comprises computer rooms of Science-Technology, Science-college and Engineering-college Buildings in Tunghai University. We used 40 clients as attackers and one node as victim. Wireshark *(version 0.99.5)* was installed on the victim to gather traffic issued by the 40 attackers, and Trinoo was employed as the attack tool which uses 27444 port to send attack packets.

In experiment 1, we issued 323,201 packets/sec by several computers in some groups, e.g., group 0 sends 18.75% of attack packets, group 1 delivers 34.38% of attack packets, etc. In experiment 2, 700,002 packets /sec were issued also by several groups.

Table 2 Packet statistics collected in experiment 1 for an event analyzer's source-IP distribution table, and the chi-square statistics

Group	Source IP	past 1^{st}-day count	...	one-week sum-up= $\sum_{1}^{7} -day-count$	current-day count	G%	10sec (baseline profile)	Past 10-sec count	N%	Chi-square for %	Chi-square for amount
G0	140.128.0.132	11,111,517	...	68,252,562	7,553,961	15.83	1,128.5	537,598	19.43	0.82	475
G1	140.128.1.85	6,169,027	...	39,800,171	4,804,121	17.41	1,241.6	712,664	31.37	11.19	698
	140.128.0.125	5,470,646	...	35,294,492	3,783,604			155,330			
G2	163.23.75.125	1,938,878	...	11,953,623	1,688,232	11.09	790.58	459,399	22.21	11.16	776
	140.128.0.123	2,326,653	...	14,344,348	1,526,847			52,248			
	...		...								
G3	140.128.201.1	686,083	...	4,513,701	1,001,435	8.72	621.93	537,427	26.64	36.82	1,184
	140.128.0.74	857,603	...	5,642,126	613,572			33,561			
	...		...								
G4	140.128.0.98	292,085	...	1,738,600	179,041	6.45	459.95	313	0.18	6.09	5001
	...		...								
G5	140.128.0.95	157,206	...	965,638	99,349	7.17	510.92	81	0.09	6.98	2605
	...		...								
G6	140.128.0.91	121,383	...	783,113	80,514	11.62	828.69	10	0.02	11.58	612
	140.128.0.78	125,003	...	886,476	87,055			10			
	...		...								
G7	140.128.0.104	59,309	...	365,655	37,593	10.85	773.87	4	0.02	10.82	498
	...		...								
G8	140.128.0.24	15,759	...	124,417	21,318	6.16	438.86	1	0.01	6.13	316
	...		...								
G9	140.128.0.88	2,659	...	15,828	1,628	1.88	134.00	1	0.01	1.86	302
	...		...								
G10	140.128.0.66	809	...	5,222	537	1.24	88.42	0	0.00	1.24	68
	...		...								
G11	140.128.0.69	434	...	2,674	275	1.27	90.54	0	0.00	1.27	-1
	...		...								
G12	140.128.0.41	435	...	2,475	76	0.31	21.93	0	0.00	0.31	-1
	...		...								
sum						100	7,129.84	2,766,800	100	106.26	

Table 3 Packet statistics collected in experiment 1 for an event-collector's source-IP accumulation table and the chi-square statistics

Source IP	port	group no	past 1^{st}-day count	...	one-week sum-up= $\sum_{1}^{7} -day-count$	current-day count	G%	10sec (baseline profile)	Past 10-sec count	N%	Chi-square for %	Chi-square for amount
140.128.0.132	27444	0	3,526,382	...	22,750,854	2,876,386	16.58	376.17	537,598	19.47	0.51	1,428
140.128.23.25	3620	1	2,056,342	...	13,266,724	1,519,150	16.09	219.36	155,330	31.44	14.64	4,883
140.128.1.85	27444	1	1,367,662	...	8,823,623	1,619,732		145.89	712,664			
163.23.70.45	3131	2	617,604	...	3,984,541	459,621	10.13	65.88	50,011	22.26	14.53	11,067
...	...	2	...	...	...	...		63.25	40,119			
163.23.75.125	27444	2	389,090	...	2,510,261	717,454		41.51	459,399			
140.128.1.125	3196	3	233,208	...	1,504,567	186,222	8.04	24.88	31,553	26.70	43.35	24,688
...	...	3	...	...	...	...		35.24	28,410			
140.128.201.1	274444	3	204,057	...	1,316,496	672,763		21.77	537,427			

Table 4 Packet statistics collected in experiment 2 for an event-collector's source-IP accumulation table (IPs of a group is represented as a tuple) and the chi-square statistics

group	past 1^{st}-day count	...	one-week sum up= $\sum_{1}^{7} -day-count$	current-day count	G%	10sec (baseline profile)	Past 10-sec count	N%	Chi-square for %	Chi-square for amount
G0	11,111,517		68,252,562	10,575,093	15.83	1,128.51	200,704	4.55	8.03	176.85
G1	11,639,673		75,094,663	13,016,199	17.41	1,241.64	400,296	9.08	3.99	321.39
G2	7,755,511		47,814,493	7,984,312	11.09	790.58	200,113	4.54	3.87	252.12
G3	5,717,355		37,614,175	6,829,380	8.72	621.93	999,183	22.66	22.28	1,605.59
G4	4,673,355		27,817,592	5,312,515	6.45	459.95	800,502	18.16	21.24	1,739.42
G5	5,030,587		30,900,414	5,501,361	7.17	510.92	804,498	18.25	17.13	1,573.61
G6	7,768,485		50,119,257	8,622,836	11.62	828.69	202,801	4.60	4.24	243.72
G7	7,591,580		46,803,820	8,219,978	10.85	773.87	600,316	13.62	0.70	774.73
G8	4,034,417		26,542,218	4,313,831	6.16	438.86	199,787	4.53	0.43	454.24
G9	1,361,490		8,104,106	1,315,083	1.88	134.00	597	0.01	1.85	3.46
G10	828,859		5,347,480	898,580	1.24	88.42	203	0.00	1.23	1.30
G11	888,178		5,475,820	891,463	1.27	90.54	0	0.00	1.27	-1.00
G12	215,901		1,326,175	201,579	0.31	21.93	0	0.00	0.31	-1.00
			431,212,775	73,682,211	100.00	7,129.84	4,409,000	100.00	86.58	

"chi-square for percentage" in table 2 established for experiment 1 is $106.26 > x^2_{12,0.05}(=21.026)$, showing that there is an attack. Also, from this table, we can realize which groups are issuing DoS/DDoS attack. From Table 3, we can accordingly realize 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 within the top 80%.

In experiment 2, the packet statistics listed in the source-IP accumulation table (Table 4) were gathered by accumulating several source-IP distribution tables. Its "chi-square for percentage" is 86.58 (>21.026). So, we can also conclude that there is a DoS/DDoS attack.

For accuracy test, we gather 100 times of normal and 100 times of attack traffic of 10 seconds. The attack densities range from 500 to 15,000 packets/sec. Table 5 shows the results. The errors (false-positives) result from the fact that at some time G% and N% had similar percentages, and were treated as normal traffic.

Table 5 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. Conclusion 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 statistical method to detect DoS/DDoS attacks. It analyzes amount and variation of packets issued by senders that send packets to a victim, and statistics of address distribution. If a hacker sends a huge amount of packets, we check to see whether or not his/her chi-square value exceeds 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*, pp. 42-51, Oct. 2002.

[2] S. Snapp et al., "DIDS (Distributed Intrusion Detection System) - Motivation, Architecture, and An Early Prototype," *Proc. of the 14th National Computer Security Conference*, pp. 167-176, 1991.

[3] J. Balasubramaniyan et al., "An Architecture for Intrusion Detection using Autonomous Agents," Technical report no. TR 98-05, Purdue University, USA, 1998.