2009 Third International Conference on Multimedia and Ubiquitous Engineering

DDoS Detection and Traceback with Decision Tree and Grey Relational Analysis

Yi-Chi Wu, Huei-Ru Tseng, Wuu Yang, and Rong-Hong Jan
Department of Computer Science
National Chiao Tung University
Hsinchu City 300, Taiwan
{ycwu, hueiru, wuuyang, rhjan}@cs.nctu.edu.tw

Abstract

As modern life becomes increasingly closely bound to the Internet, network security becomes increasingly important. Like it or not, we all live under the shadow of network threats. The threats could cause leakage of privacy and/or economic loss. Among network attacks, the DDoS (distributed denial-of-service) attack is one of the most frequent and serious.

In a DDoS attack, an attacker first breaks into many innocent computers (called zombies) by taking advantages of known or unknown bugs and vulnerabilities in the software. Then the attacker sends a large number of packets from these already-captured zombies to a server. These packets either occupy a major portion of the server's network bandwidth or they consume much of the server's time. The server is then prevented from conducting normal business operations.

In order to mitigate the DDoS threat, we design a system to detect DDoS attacks based on a decision-tree technique and, after detecting an attack, to trace back to the approximate locations of the attacker with a traffic-flow pattern-matching technique. We conduct our experiment on the DETER system. According to our experiment results, our system could detect the DDoS attack with the false positive ratio about 1.2% – 2.4%, false negative ratio about 2% – 10% with different kind of attack, attack sending rate and find the attack path in traceback with the false negative rate 8% – 12% and false positive rate 12% – 14%.[1]

1. Introduction

With the proliferation of computer networks come many kinds of network attacks. Among them, the DDoS attacks [1][2][3][4] have caused serious economic loss. Therefore, effective and efficient protection systems are urgently needed. Denial-of-service attacks, as the term suggests, attempt to deny legitimate users the services that the servers provide.

1. The work reported in this paper is partially supported by National Science Council (NSC), Taiwan, Republic of China, under grants NSC 96-2628-E-009-014-MY3, NSC 97-2218-E-009-029, NSC 97-2623-7-036-001-D, and NSC 97-2221-E-009-048-MY3, NSC 97-2221-E-009-049-MY3.

Because an attacker could modify the source IP addresses in the packets (i.e. IP spoofing), tracing back the origin of an attack becomes very difficult. We design a system that detects DDoS attacks quickly and traces back the origins of DDoS attacks quite accurately. The characteristics of our system include: (1) there is no need to modify existing protocols (e.g., TCP/IP); (2) the setup procedures on routers are simple; (3) the system can accommodate novel attacks in the future; (4) the system can fit any network topology; (5) the traceback procedure is efficient.

In this paper, we focus on the flooding-based attack aiming at layer 3/layer 4 in the OSI 7-layer model and apply the decision tree (C4.5) technique to detect abnormal traffic flow [4]. Then we use a novel traffic pattern matching procedure to identify the traffic flow that is similar to the attack flow and, based on this similarity, to trace back the origin of an attack. The attack path reconstruction is accomplished by the *protection agent* and the *sentinels*.

There are several DDoS detection and traceback mechanisms. These can be divided into two categories depending on the locations of the DDoS attack detection systems. One is victim based, in which the detection system is deployed close to the victim. The other is source based, in which the detection system is placed close to the attack source. Victim-based detection makes use of machine learning techniques [5], statistical models [6], etc., to identify the traffic patterns under attack. Source-based detection attempts to deploy the detection systems as close to the attacker as possible [7].

Traceback is also an important defense against of DDoS attacks. Due to IP spoofing, the source IP address in a packet is of little use in traceback. There are several IP traceback methods, including link testing [8], packet marking, and ICMP traceback.

The rest of this paper is organized as follows. In section 2, we introduce the architecture of our system. In section 3, our proposed detection and traceback method will be introduced. In section 4, there will be the experiment results, which indicate that our proposed system is capable of detecting the attacks and tracing them back with high accuracy. In the last section, there will be conclusion and future work.

978-0-7695-3658-3/09 $25.00 © 2009 IEEE
DOI 10.1109/MUE.2009.60

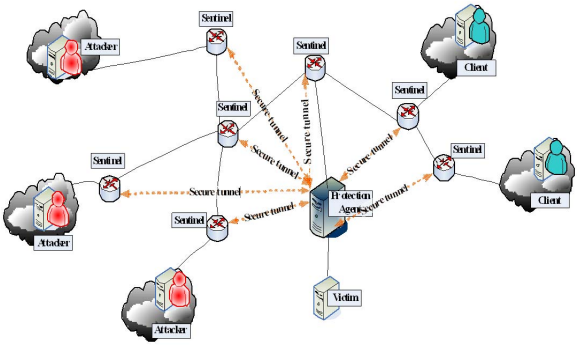

Figure 1. Overall organization of our system.

2. Proposed System

In this section we will describe our detection and traceback system. It includes an artificial intelligence-based (AI-based) classifier for DDoS detection and a traffic-flow pattern matcher for comparing traffic signatures and for tracing back DDoS attacks.

2.1. System Architecture

Our system consists of two components: *protection agents* and *sentinels*. A protection agent is deployed at the victim site for the detection purpose and the sentinels are deployed at all the routers for the traceback purpose. The overall organization is shown in Figure 1. The links between the protection agent and the sentinels are secured tunnels, which make use of port forwarding in SSH-2 (secure shell protocol version 2) for preventing man-in-the-middle attacks.

2.2. System Modules

In this subsection, we will introduce the components within the protection agent and sentinels.

2.2.1. Protection Agent. The protection agent is the control center of the entire system. The DDoS attack detection and attack path reconstruction are all handled in the protection agent. A protection agent consists of four components: a packet aggregator (to aggregate the traffic signatures), a message manager (to construct the SSH-tunnel and handle communication between the protection agent and sentinels), a DDoS attack detection module, and a traceback module. The DDoS attack detection module includes the decision tree and rules. The message manager resides in the traceback module; the traceback module handles attack path reconstruction. The procedure of the protection agent is shown below.

1 Obtain the signature of the current traffic flow.

Table 1. Format of a traffic signature.

Attributes	Value
1.Incoming packet count per t minute(s)	Numeric
2.Incoming octets per t minute(s)	Numeric
3.# of incoming TCP packets per t minute(s)	Numeric
4.# of incoming UDP packets per t minute(s)	Numeric
5.# of incoming ICMP packets per t minute(s)	Numeric
6.# of incoming unknown-protocol packets per t minute(s)	Numeric
7.# of incoming IP addresses / # of outgoing IP addresses	Numeric
8.Outgoing packet count per t minute(s)	Numeric
9.Outgoing octets per t minute(s)	Numeric
10.# of outgoing TCP packets per t minute(s)	Numeric
11.# of outgoing UDP packets per t minute(s)	Numeric
12.# of outgoing ICMP packets per t minute(s)	Numeric
13.# of outgoing unknown-protocol packets per t minute(s)	Numeric
14.# of incoming TCP SYN packets / # of incoming TCP ACK packets	Numeric
15.# of incoming IP addresses / # of incoming packets per t minute(s)	Numeric
16.Time interval	Bed-time, morning, afternoon, night

2 The detection module determines if an attack is going on based on the current traffic signature.

3 If there is no attack, the agent stores the traffic signature into the repository.

4 If there is an attack, then

- 4.1 The agent issues a traceback command to the upstream sentinel.
- 4.2 Wait until enough connection information (which contains the IP addresses of the two ends of the link and the distances from the victim) is collected.
- 4.3 Construct the attack path with the collected connection information.

5 Go to step 1.

2.2.2. Packet Aggregator. The packet aggregator computes a *traffic signature* based on all the packets passing through. The traffic signature is used for detection and traceback. With the help of pcap, our system captures all incoming and outgoing packets. For each packet, the packet header from layer 3 to layer 4 is extracted for cross-layer monitoring. The header information is used to compute a traffic signature, whose format is shown in Table 1. Our system generates one traffic signature per minute.

The traffic signatures are stored in the traffic signature repository with timestamps of the packet arriving time. A Bloom filter [9] is used to reduce the memory overhead while collecting the IP addresses. The Bloom filter computes k (which is the number of hash functions used in the bloom filter) distinct digests for each IP address with independent hash functions, and uses the n-bit results to index into a 2^n- bit array. An example of a Bloom filter is depicted in Figure 2. We implemented two basic hash functions, SAX and SDBM, as the default hash functions in the Bloom filter. The flow chart for packet processing is depicted in Figure

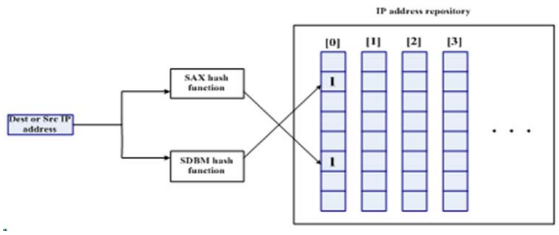

Figure 2. Bloom filter.

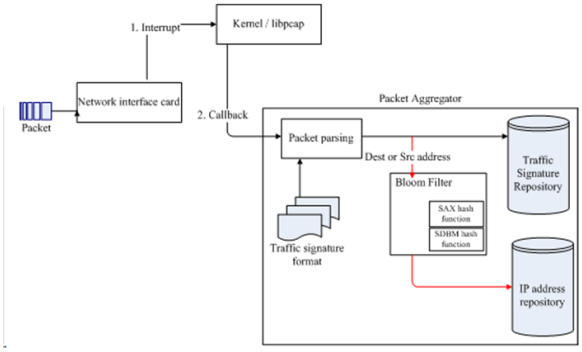

Figure 3. Packet aggregator.

Table 2. Format of simplified traffic signature.

Attributes	Value
1.# of incoming TCP packets per t minute(s)	Numeric
2.# of incoming UDP packets per t minute(s)	Numeric
3.# of incoming ICMP packets per t minute(s)	Numeric

3. With the help of the Bloom filter, we could reduce the 32-bit IP address into 2 bits.

2.2.3. n-Hop Sentinels. The n-hop sentinel is located in a router. The "n" represents the number of hops from the victim. A sentinel aggregates the headers of incoming packets into a simplified format of traffic signature, shown in Table 2, which is similar to a traffic signature.

A sentinel collects the packets and transforms them into traffic signature. When receiving a traceback command, the message manager would identify the attack type in the command and perform the grey relational analysis to find the possible entrances of attack traffic.

1 Obtain the signature of the traffic flow.
2 Upon receiving a traceback command, a sentinel modifies and forwards the command to upstream sentinels that are the possible entrances of attack traffic identified by the traffic-pattern matching module.
3 Send the connection information back to the protection agent.

3. DDoS Detection and Traceback Mechanism

3.1. DDoS Detection

It is reasonable to assume that the attack traffic would be different from the normal traffic in some aspects. We build a base-line traffic profile from the normal network traffic. Whenever the network traffic deviates from the base-line profile *significantly*, an attack is alarmed. We adopt a decision-tree classifier [10] to classify network traffic. The advantage of the decision-tree classifier is its efficiency in both generalization and new attack detection. A decision tree consists of leaf nodes representing classes and non-leaf nodes that specify tests to be carried out on a particular attribute. The construction of decision trees is based on training data. Then the classifier is used to new data.

We adopt the C4.5 [11] algorithm to construct the decision tree. C4.5 chooses the attribute as the splitting criterion according to the entropy-based gain ratio in order to overcome the over-fitting problem. First, C4.5 defines $info(T)$ in equation (1). $info(T)$ represents the entropy of the training data set T and represents the probability that one random instance from T belongs to a class C_j (there are four classes in our system: Normal, TCP SYN attack, UDP attack, and ICMP attack and one traffic signature aggregated per 1 minute would be considered as one instance in our system).

$$info(T) = -\sum_{j=1}^{k}\left[\frac{|T_j|}{|T|} \times log_2\left(\frac{|T_j|}{|T|}\right)\right] \quad (1)$$

Then the gain information for an attribute is defined in equation (2). $gain(X)$ measures the quantity of information that is gained by partitioning T according to the attribute X (we treat the format of traffic signature defined in Table 1 as the attributes in our system).

$$gain(X) = info(T) - \sum_{i=1}^{n}\left[\frac{|T_i|}{|T|} \times info(T_i)\right] \quad (2)$$

where T_i represents the number of instances in the specific attribute. Then the gain ratio is defined in equation (3).

$$gain\_ratio(X) = \frac{gain(X)}{-\sum_{i=1}^{n}\left[\frac{|T_i|}{|T|} \times log_2\left(\frac{|T_i|}{|T|}\right)\right]} \quad (3)$$

The attribute with the largest gain ratio is selected as the splitting criterion in the decision tree. Based on the selected attribute, the training data set is then divided into several subsets. Another attribute is similarly selected and each subset is further split. The splitting procedure is repeated until all the data in a subset belong to the same class or the gain ratios of all the attributes are the same. The construction procedure is summarized as follows:

1 Select the attribute with the largest gain ratio as the splitting criterion, and create a branch for each possible value of the selected attribute.
2 Divide the instances in the training data set into subsets according to the selected attribute.
3 Repeat steps 1 and 2 for each branch.

In our implementation, we define four classes —normal, TCP SYN flooding, UDP flooding, ICMP flooding—and 16 attributes derived from the traffic signatures. A decision tree is then constructed from the training data set. According to the decision tree, the incoming traffic is classified.

3.2. Traceback

When the protection agent detects an attack, it raises an alarm to the traceback module. Because the source IP address in a packet could be easily spoofed, it cannot be used for traceback. Instead, we make use of traffic-flow pattern matching for traceback. Our objective is to find the routers where the attack traffic first enters the network. Starting from the victim, we attempt to discover the routers on the attack path one by one, until we reach the entry points of the attack traffic. For each router, we identify the incoming link on which the *incoming* traffic is most similar to the *outgoing* attack traffic. Then that link is deemed to be on the attack path.

In order to determine the similarity of the traffic on the communication links, we make use of traffic-flow pattern matching [12]. There are two separate procedures in pattern matching: *trend-pattern matching* and *volume-pattern matching*. Traffic pattern matching is done in sentinels and the results are collected by the protection agent, which will construct the attack paths.

3.2.1. Attack Edge Determination. When a DDoS attack was detected, the traceback module will wait for sentinels to aggregate the traffic signatures. Then the traceback module puts the attack traffic signatures aggregated during the attack and the timestamp when the attack was detected into the traceback command.

The traceback module in the protection agent issues a traceback command to the upstream sentinel, which is referred to as the *1-hop sentinel*. When the 1-hop sentinel receives the traceback command, it searches the traffic signature repository for every network interface card (NIC) to retrieve the traffic signatures with the appropriate attack type and the aggregated timestamp that matches the timestamp in the traceback command. Afterwards, the sentinel applies the traffic-flow pattern matching algorithm to identify the set of NICs that are the possible entrances the attack traffic may come from. If a router is equipped with n network interface cards, there will be $2^n - 2$ different combinations (the two cases—no NIC and all NICs—are ignored). And if $n = 1$, then this only one NIC will be considered as the entrance of attack traffic. After identifying the suspicious entrances of the attack traffic, the sentinel would send the connection information (which includes IP addresses of the two ends of the link and their distances from victim) to the protection agent through a SSH tunnel. In this way, the protection agent could receive the links that the attack traffic might pass through.

1. **begin** *Modified attack edge sampling:*
2. *find the set of NICs whose traffic is closest to the evidence in the traceback command;*
3. *If command.ip == NULL then*;
4. *fill the IP address of the suspicious NIC in the "ip" field of the command;*
5. $else$
6. *edge_info.start_ip := command_ip;*
7. *edge_info.end_ip := ip address of the suspicious NIC;*
8. *edge_info.distance := command.distance +1;*
9. *send edge_info back to the protection agent;*
10. $endif$
11. *command.start := the ip address of the suspicious NIC;*
12. *increment the command.distance;*
13. *fill the traffic through the suspicious NICs in the "evidence" field as new evidence;*
14. *forward the modified command to the suspicious NICs;*
15. **end**

Figure 4. Algorithm for determining the attack edges.

After sending back the connection information, the sentinel puts the traffic signatures aggregated by the suspicious NIC into the traceback command as new evidence and forwards this modified command to the upstream sentinel. When the upstream sentinel receives the traceback command, it repeats the same procedure until the entrance router is reached. Figure 4 illustrates the algorithm of the edge of attack path sampling. After all the connection information is collected, the attack paths could be reconstructed.

3.2.2. Traffic-Flow Pattern Matching. In a DDoS attack, the attack traffic enters the network from multiple routers and flows to a single victim. The communication links that the attack traffic passes through form a tree (under normal routing) with the victim as the root and the entrances as the leaves. Our traceback method starts from the victim and identifies the routers on the tree one by one. Each sentinel will find the upstream routers on the tree.

The major problem in our traceback method lies in identifying the upstream routers of attack traffic in the sentinels. Therefore, the aim of traffic-flow pattern matching is to identify the subset of NICs on a router whose collective incoming traffic has similar signature as the attack traffic that goes out of that router. We apply two kinds of pattern

matching techniques which measure both the *trend* and the *volume* of network traffic.

- **Trend-Pattern Matching:** Trend-pattern matching is based on the assumption that the DDoS attack traffic should dominate the *change* in the outgoing traffic from a router. Hence, we need to characterize the traffic trend quantitatively and determine if they are similar.
  Unlike other systems which compare traffic signatures with conventional statistical methods, grey relational analysis (GRA) [13][14] is used in our system. GRA is applicable to a small sample size. Because the duration of the observation window[2] is quite short, the resulting sample size is quite small. There are three pre-processing steps for the GRA:
  i Grey relational maximizing operation
  ii Grey relational coefficient computation
  iii Grey relational grade computation
  During the observation window, the 1-hop sentinel computes a sequence of traffic signatures (one per minute) for each combination of NICs. The maximizing operation (equation (4)) is then applied to normalize the signatures. The purpose of the maximizing operation is to diminish the magnitude of sequences and make them comparable.

$$y(k) = \frac{x(k)}{x_{max}} \tag{4}$$

  where the original signature sequence is $x = \{x(1), x(2), \ldots, x(n)\}$ and the normalized sequence is $y = \{y(1), y(2), \ldots, y(n)\}$.
  There is a sequence of signatures for each combination of NICs on a router. Among the many sequences $y_1, y_2, \ldots$ etc., we wish to find the combination of NICs whose sequence of signatures is most similar to the sequence y_0 of the signatures of the attack traffic. We use equation (5) [15] to compute *grade* $r(y_0, y_i)$, for each sequence y_i.

$$r(y_0, y_i) = \left(\frac{\Delta_{max} - \overline{\Delta_{0i}}}{\Delta_{max} - \Delta_{min}} \right) \tag{5}$$

  where $\Delta_{min} = \min_{\forall i} \min_{\forall k} \Delta_{0i}(k) = \min_{\forall i} \min_{\forall k} |y_0(k) - y_i(k)|$,
  $\Delta_{max} = \max_{\forall i} \max_{\forall k} \Delta_{0i}(k) = \max_{\forall i} \max_{\forall k} |y_0(k) - y_i(k)|$
  and $\overline{\Delta_{0i}} = \frac{1}{n} \sum_{k=1}^{n} \Delta_{0i}(k)$
  $r(y_0, y_i)$ may be interpreted as the similarity between the sequences y_0 and y_i. If $r(y_0, y_1) > r(y_0, y_2)$, we may conclude that the sequence y_1 is more similar to the sequence y_0 than the sequence y_2.

2. The observation window is the period during which the protection agent collects attack traffic after a DDoS attack is detected. It is usually 2-5 minutes in our system.

- **Volume-Pattern Matching:** Trend-pattern matching considers only the similarity of two sequences but not their magnitude (here magnitude means the volume of traffic). *Volume-pattern matching* will compare the magnitudes of two sequences. We use equation (6) to compute a *volume coefficient* g_i for each sequence x_i.

$$g_i = \frac{\sum_{k=1}^{n} \sqrt{x_0(k) x_i(k)}}{\sum_{k=1}^{n} x_0(k)} \tag{6}$$

3.2.3. Traceback Command Forwarding Policy. The grade $r(y_0, y_i)$ represents the similarity in shape of the two sequences y_0 and y_i while the volume coefficient g_i represents the similarity in volume of the two sequences. When the grade $r(y_0, y_i)$ is greater than a selected threshold T_{trend}, we claim that the two sequences have the same shape. Similarly, when the volume coefficient g_i is greater than a certain threshold T_{vol}, we claim that two sequences have the same volume. In our system, we use $T_{trend} = 0.8$ and $T_{vol} = 0.9$. This would reduce the false positive/negative ratios in our experiment.

We consider only the sequences x_i for which $r(y_0, y_i) > T_{trend}$ and $g_i > T_{vol}$. Among these sequences, we choose the sequence with the largest $r(y_0, y_i)$. The subsets of NICs corresponding to the chosen sequence are deemed as the entrances for the attack traffic to enter the router. When there is no sequence for which $g_i > T_{vol}$, we claim that all the NICs are the entrances for the attack traffic to enter the router. Otherwise, the router is not on the attack path and the sentinel on the router will stop forwarding the traceback command.

4. Experiment Design and Results

4.1. Simulation Design

We verified the performance of the proposed detection and traceback system on the DETER test-bed [16]. The DETER test-bed provides users an environment to emulate the real-world network traffic with an easy-to-use web interface and various tools, such as SEER, a benchmark for DDoS defense mechanism. There are three major components in a DDoS attack experiment: topology design, legitimate traffic (background traffic), and attack traffic.
Topology Design: In our experiment, there are 5 zombie attackers, 20 routers, and 10 clients which perform common web browsing. The network topology is generated with the Waxman algorithm [17], which is shown in Figure 5.
Legitimate Traffic Generation: The background traffic (i.e., normal traffic, without attacks) is generated with harpoon [18] from the actual trace data collected at the computing center of Department of Computer Science in National Chiao Tung University. The machine is a web page server in the center.

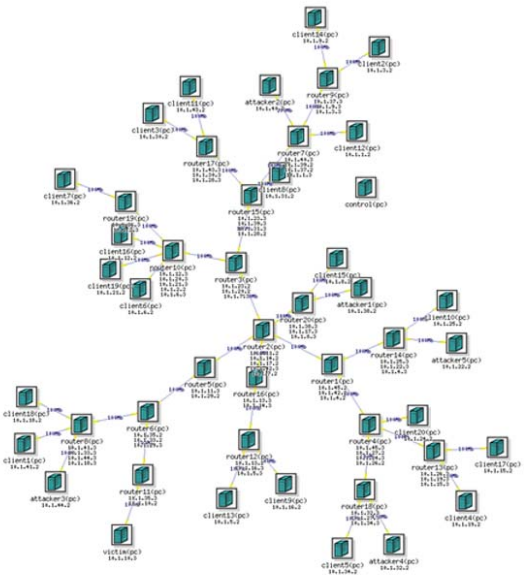

Figure 5. Experiment topology design.

Table 3. Attack scenario in training data.

TCP SYN flood	UDP flood	ICMP flood
150 pkt/per sec Pkt. size: 66 bytes	150 pkt/per sec Pkt. size: 256 bytes	150 pkt/per sec Pkt. size: 256 bytes

Table 4. Attack scenario for evaluating purpose.

Scenario 1:		
TCP SYN flood	**UDP flood**	**ICMP flood**
250 pkt/per sec Pkt. size: 66 bytes	250 pkt/per sec Pkt. size: 256 bytes	250 pkt/per sec Pkt. size: 256 bytes
Scenario 2:		
TCP SYN flood	**UDP flood**	**ICMP flood**
150 pkt/per sec Pkt. size: 66 bytes	150 pkt/per sec Pkt. size: 256 bytes	150 pkt/per sec Pkt. size: 256 bytes
Scenario 3:		
TCP SYN flood	**UDP flood**	**ICMP flood**
70 pkt/per sec Pkt. size: 66 bytes	70 pkt/per sec Pkt. size: 256 bytes	70 pkt/per sec Pkt. size: 256 bytes

Table 5. Situation analysis in detection.

	Detection Result	
Actual Situation	Attack	Normal
Attack	A	B
Normal	C	D

'A' is the number of attack signatures that are successfully and correctly detected by the protection agent; 'B' is the number of attack signatures that the protection agent failed to detect; 'C' is the number of reported attack signatures while there is actually no attack; and 'D' is the number of normal traffic signatures that are recognized as normal (that is, not identified as an attack).

Experiment Scenarios: The background traffic was collected from June 25, 2008 (Wednesday) midnight to June 27, 2008 (Friday) midnights (48 hours in total), which is divided into two groups: the traffic on June 25 (denoted as *data25*) is used as training data set while the traffic collected on June 26 (denoted as *data26*) for testing purpose. Each day is further divided into four periods: bed-time (0 am - 7 am), morning (7 am -12 pm), afternoon (12 pm -18 pm), and evening (18 pm - 24 am).

There are four iterations in our experiment. The first two iterations constitute the training phase while the last two the testing phase. First we feed the simulation environment with data25. We obtain 1440 signatures (one per minute) for the background traffic. Second, we feed the simulation with data25 plus randomly generated attack traffic. We obtain another 1440 signatures and these signatures with attack traffic would be denoted as attack traffic signatures. The two sets of signatures are used to build the decision tree with the C4.5 algorithm. Third, we feed the simulation with data26. The resulting 1440 signatures are used to calculate the false positive ratio. Finally, we feed the simulation with data26 and randomly generated attack traffic. The resulting 1440 signatures will be used to calculate the false negative ratio and the false classification ratio.

Attack Traffic Generation: The attack traffic is randomly generated with the SEER tool. We tested three kinds of attacks: TCP SYN flood, UDP flood, and ICMP flood. In order to simplify the experiment, at most one attack is underway at any time. Each attack lasts for 1 hour in the training phase and for 12 minutes during the testing phase. The amount of attack traffic for each attack during the training phase is shown in Table 3. The amount of attack traffic during the testing phase is shown in Table 4. The testing phase is repeated three times, each with different attack traffic.

4.2. Performance Evaluation

4.2.1. Performance Metrics.

- **Performance Metrics for DDoS Detection:** In detecting DDoS attacks, we focus on four metrics: FNR (false negative ratio), FPR (false positive ratio), FCR (false classification ratio), and Detection Latency. The definitions of FNR and FPR are listed as equation (7) and equation (8) according to Table 5.
  Table 6 defines the FCR for different attacks. In TCP SYN flooding attack, the false classification represents the ratio between the number of TCP SYN flooding attack traffic signatures that are detected as UDP flooding attack signatures or ICMP flooding attack signatures and the total number of TCP SYN flooding attack traffic signatures. The false classification ratio in UDP flooding attack and ICMP flooding are deduced as the same way. Detection latency represents the average number of time slots needed to detect the attack when the attack was launched in our experiment.

Table 6. The false classification ratio.

	Equation
TCP SYN flood	$\frac{N_{i=ICMP}+N_{i=UDP}}{R_{j=TCP}}$
UDP flood	$\frac{N_{i=ICMP}+N_{i=TCP}}{R_{j=UDP}}$
ICMP flood	$\frac{N_{i=TCP}+N_{i=UDP}}{R_{j=ICMP}}$

N_i represents the number of attack traffic signatures that the protection agent detected as attack i; R_j represents the number of attack traffic signatures actually generated by attack j.

Table 7. Situation analysis in traceback.

	Report Result	
Actual Situation	Attack Path	Normal Path
Attack Path	E	F
Normal Path	G	H

'E' is the number of attack edges that are successfully and correctly reported by sentinels; 'F' is the number of attack edges the sentinels failed to identify; 'G' is the number of edges that are not on the attack path but are mistakenly identified as attack edges by the sentinels; and 'H' is the total number of edges that are not on the attack path and are recognized as normal.

$$FNR = \frac{B}{A+B} \tag{7}$$

$$FPR = \frac{C}{C+D} \tag{8}$$

According to Table 5, the FNR (see equation (7)) represents the ratio between the number of attack traffic signatures that are not detected and the total number of attack traffic signatures. FPR (see equation (8)) represents the ratio between the number of normal traffic signatures that are claimed as attack signatures and the total number of normal signatures.

- **Performance Metrics for DDoS Traceback:** In DDoS traceback performance evaluation, we define the MNER (misidentified normal edge ratio) and MAER (misidentified attack edge ratio) as our metrics for traceback. According to Table 7, MNER (see equation (9)) represents the ratio between the numbers of normal edges but are claimed as attack edges by the sentinels and the total number of normal edges. MAER (see equation (10)) represents the ratio between the number of attack edges that are not found by the sentinels and the total number of attack edges.

$$MNER = \frac{G}{G+H} \tag{9}$$

$$MAER = \frac{F}{E+F} \tag{10}$$

4.2.2. Performance Evaluation.

- **Performance of DDoS Detection:** Figure 6 depicts the false positive ratios for the four periods in a day. The result indicates that the false positive ratio ranges from 1.2% (bed time) to 2.4% (morning). In [5], the false positive ratio ranges from 1% to 8% depending on the background traffic. However, it is not clear the amount of attack traffic in [5]. In D-WARD [19], the false positive ratio (which is called *false alarm*) is about 2%. However, it is clear about the amount of attack and normal traffic.

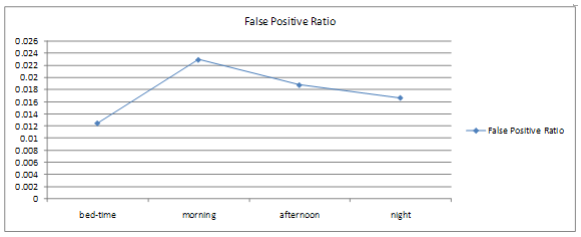

Figure 6. False positive ratio in DDoS detection.

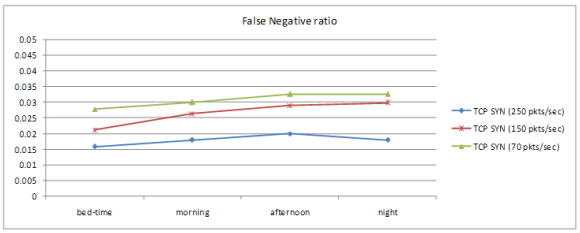

Figure 7. False negative ratio in TCP SYN flooding.

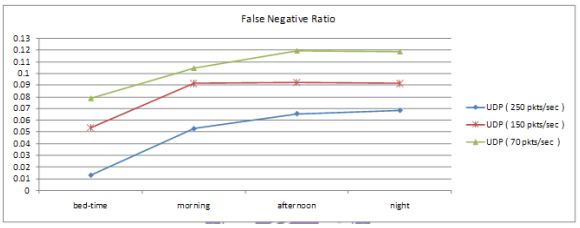

Figure 8. False negative ratio in UDP flooding.

Figure 7, Figure 8, and Figure 9 depict the false negative ratios of TCP SYN flooding, UDP flooding, and ICMP flooding, respectively. Because the sending rates in the ICMP flooding attack and the UDP flooding attack are the same, the results in ICMP and UDP flooding attacks are similar. When the attack rate is 150 packets per second (note that in the training phase the attack rate is 150 packets per second), the false negative ratio ranges from 5% to 10% for UDP and ICMP flooding. The false negative ratio is 2% to 3% for the TCP SYN flooding.

Figure 10, Figure 11, and Figure 12 depict the results in the false classification ratios. The results show that the false classification ratio for the TCP SYN attacks is lower than that for ICMP and UDP flooding attacks. Nearly 40% to 50% of ICMP attacks may be mistaken as UDP attacks. Similarly, nearly 40% to 50% of UDP attacks may be mistaken as ICMP attacks. On the other hand, TCP SYN attacks are seldom mis-classified.

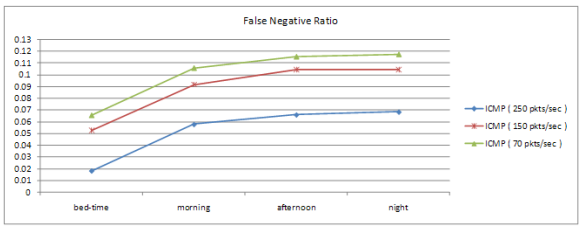

Figure 9. False negative ratio in ICMP flooding.

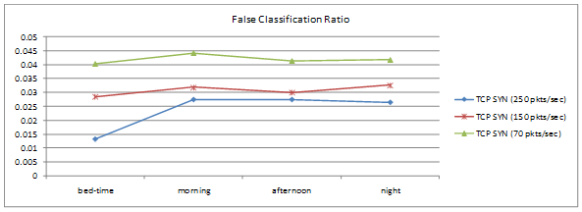

Figure 10. False classification ratio in TCP SYN flooding.

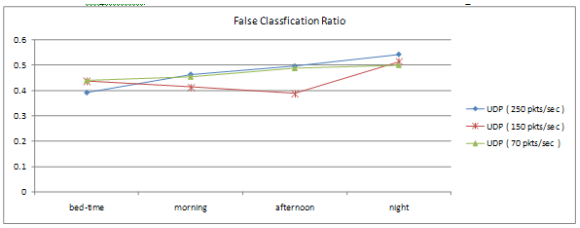

Figure 11. False classification ratio in UDP flooding.

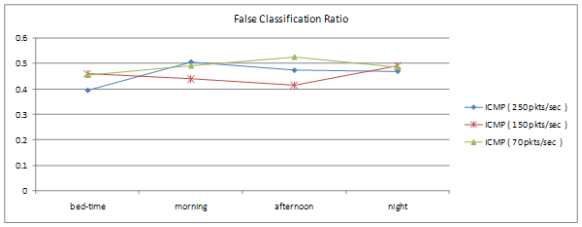

Figure 12. False classification ratio in ICMP flooding.

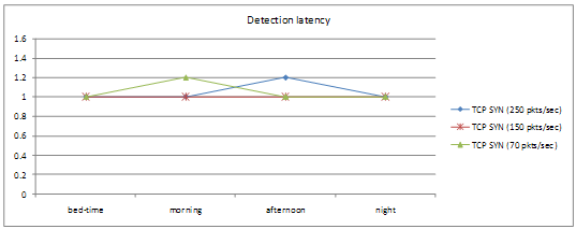

Figure 13. Detection latency in TCP SYN flooding.

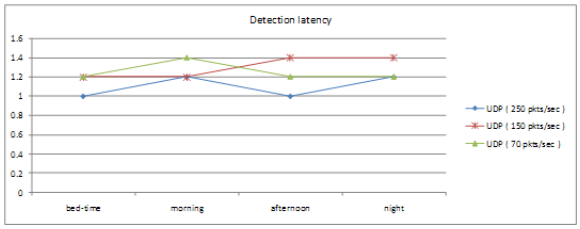

Figure 14. Detection latency in UDP flooding.

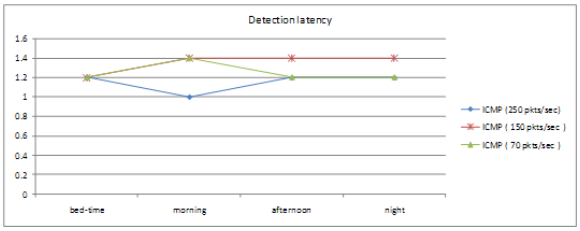

Figure 15. Detection latency in ICMP flooding.

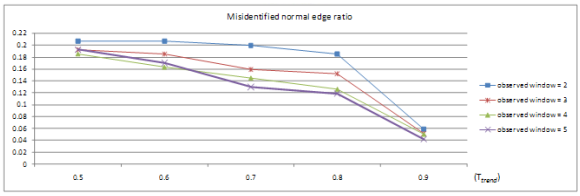

Figure 16. Misidentified normal edge ratio (MNER) in traceback.

Another important issue in DDoS detection is the *detection latency*, that is, how soon the system will claim an attack after the attack traffic reaches the victim. In our system, time is sliced into 1-minute slots. According to the results in Figure 13, Figure 14, and Figure 15, our system could claim an attack within 1 to 1.4 minutes under different attack rates.

- **Performance of Attacker Traceback:** When reconstructing the attack paths, it is possible to mistake an edge that is *not* on the attack path as an attack edge and vice versa. Figure 16 and Figure 17 show MNER and MAER with different observed windows and different trend-pattern thresholds. Remember the *observed window* is the amount of time the sentinels collect traffic data after an attack is claimed. Figure 17 shows that MAER is almost a constant while $T_{trend} < 0.9$ regardless of the observed window. The results also verify that the grey relational analysis is suitable for small sample space (size of sample space < 30).
When we keep MAER low (that is, the $T_{trend} < 0.9$), the lowest MNER is around 12% – 18.5% (from Figure 16). Furthermore we enforce an observed window for at least 3 minutes, MNER falls between 12% to 14%. In [20], MNER is 17% – 19% in old *iTrace* model and 6% in new proposed model under different network traffic. MNER in Figure 16 is less than 10% but that system makes use of a modified probability packet marking mechanism, which involves many other issues.

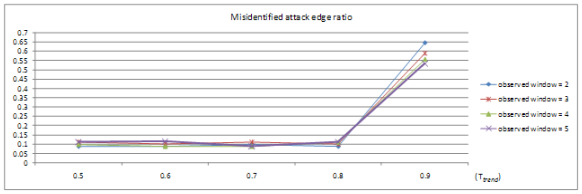

Figure 17. Misidentified attack edge ratio (MAER) in traceback.

5. Conclusions

In this paper, we propose a DDoS defense system, which includes attack detection by decision tree and attacker traceback with traffic-pattern matching. Our system is based on the observation that the network traffic under DDoS attack would differ from the traffic in normal situation. We apply the decision tree (C4.5) generating algorithm to construct the classification model and detect abnormal traffic flow. In traceback phase, we use a novel traffic pattern matching procedure, which is based on grey relational analysis, to identify the traffic flow that is similar to the attack flow and, based on this similarity, to trace back the origin of an attack. The attack path reconstruction is then accomplished by the protection agent and the sentinels.

We conduct our experiment on the DETER system. According to our experiment results, our system could detect the DDoS attack with the false positive ratio about 1.2% – 2.4%, false negative ratio about 2% – 10% with different attacks and attack sending rates and find the attack path in traceback. The misidentified attack edge ratio is about 8% – 12% and misidentified normal edge ratio about 12% – 14%. The result indicates that our proposed system is capable of detecting the attacks and tracing them back with high accuracy and within a short time.

References

[1] P. Zaroo, "A survey of DDoS attacks and some DDoS defense mechanisms," *Technical Report, Lectures Notes for Advanced Information Assurance (CS626)*, Computer Science Department, Purdue University, 2002.

[2] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Computer Communications Review*, vol. 34, no. 2, pp. 39-54, Apr. 2004.

[3] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: Classification and state-of-the-art," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 44, no. 5, pp. 643-666, Apr. 2004.

[4] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the DoS and DDoS problems," *ACM Computing Surveys*, vol. 39, no. 1, 2007.

[5] S. Noh, C. Lee, K. Choi, and G. Jung, "Detecting distributed denial of service (DDoS) attacks through inductive learning," In *Proceedings of IDEAL'03*, Mar. 2003.

[6] B. Lim and Md. S. Uddin, "Statistical-based SYN-flooding detection using programmable network processor," In *Proceedings of ICITA'05*, vol. 2, pp. 465-470, Jul. 2005.

[7] J. Mirkovic and P. Reiher, "D-WARD: A source-end defense against flooding denial-of-service attacks," *IEEE Trans. on Dependable and Secure Computing*, vol. 2, no. 3, pp. 216-232, Jul.-Sep. 2005.

[8] H. Burch, "Tracing anonymous packets to their approximate source," In *Proceedings of 14th USENIX Conference on System Administration*, pp. 319-328, Dec. 2000.

[9] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, pp. 422-426, Jul. 1970.

[10] L. Rokach and O. Maimon, "Top-down induction of decision trees classifier – a survey," *IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 35, no. 4, pp. 476-487, Nov. 2005.

[11] J. R. Quinlan, "C4.5: Programs for machine learning," Morgan Kaufmann Publishers, San Mateo, CA., 1993.

[12] G. Mansfield, K. Ohta, Y. Takei, N. Kato, and Y. Nemoto, "Towards trapping wily intruders in the large," *Computer Networks*, vol. 34, no. 4, pp. 659-670, Oct. 2000.

[13] J. L. Deng, "Introduction to grey system theory," *The Journal of Grey System*, vol. 1, no. 1, pp. 1-24, Nov. 1989.

[14] Y. Lin and S. Liu, "A historical introduction to grey systems theory," In *Proceedings of 2004 IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, pp. 2403-2408, Oct. 2004.

[15] K. H. Hsia, M. Y. Chen, and M. C. Chang, "Comments on data pre-processing for grey relational analysis," *Journal of Grey System*, vol. 7, no. 1, pp. 15-20, Jun. 2004.

[16] T. Benzel, R. Braden, D. Kim, C. Neuman, A. Joseph, and K. Sklower, "Experience with DETER: A testbed for security research," In *Proceedings of TRIDENTCOM'06*, Mar. 2006.

[17] B. M. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 1617-1622, Dec. 1988.

[18] J. Sommers, H. Kim, and P. Barford, "Harpoon: A flow-level traffic generator for router and network tests," *ACM SIGMETRICS Performance Evaluation Review*, vol. 32, no. 1, Jun. 2004.

[19] J. Mirkovic, G. Prier, and P. Reiher, "Source-end DDoS defense," In *Proceedings of 2nd IEEE International Symposium on Network Computing and Applications (NCA'03)*, pp. 171-178, Apr. 2003.

[20] A. Izaddoost, M. Othman, and M. F. A. Rasid, "Accurate ICMP traceback model under DoS/DDoS attack," In *Proceedings of ADCOM'07*, pp. 441-446, Dec. 2007.