Intrusion Detection, Forecast and Traceback Against DDoS Attacks

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

Abstract

Nowadays, DDoS is one of the most troublesome attacks. Attackers often penetrate innocent routers and hosts to make them unwittingly participate in such large-scale attacks acting as zombies or reflectors. Also, the Internet consists of autonomous network management units. Organizing these units is helpful in detecting DDoS attacks if several adjacent or nearby network management units could collaboratively guard and protect their important surrounded neighbors. In this article, we propose an Intrusion Detection, Forecast and Traceback System (IDeFT) based on united defense environment. First, a detection system that is able to detect two types of attacks, logical and DoS/DDoS, is developed. Logical attacks are recognized by neural networks. DDoS, distributed reflective DoS and what role a host/router plays in the two types of attacks are identified by the CUSUM algorithm. A hash-based intrusion tracer is also deployed to trace back to malicious clients. A forecasting model which plays the role as a proactive intrusion prevention system monitors network forwarding traffic to forecast malicious behaviors previously for its neighbor unit. Network management units with the properties of regional cooperation and autonomy can carry their network security to a higher achievement level.

Keywords: DDoS, DRDoS, intrusion detection, intrusion traceback, CUSUM

I. Introduction

Networks have dramatically changed part of human's daily activities particularly on how we communicate and how we learn and conduct business. Unfortunately, while enjoying Internet convenience, we also have to face network security problems. Intruders from anywhere may attack a site at any time causing nearly irreparable damage. Generally, defending a system against attacks is crucial. Intrusion detection systems (IDSs) and firewalls are usually deployed to protect network systems. But when facing large-scale attacks, some of them often lose their protective capabilities [1,2]. An IDS, acting as a notifier to notify network administrator and sometimes helping trace back to hackers, must be lightweight in computation so as to respond quickly, and preferably, immediately [1].

The design of IDS often focuses on detecting efficiency and accuracy. A variety of detection techniques, e.g., detecting protocol and traffic anomalies [3], has been proposed to improve detection performance. Some have exploited artificial intelligence [4], such as rule-based systems or neural networks, to make use of system learning so that unknown intrusions can be identified without manually updating detection signatures or patterns, whereas others have further promoted IDSs to intrusion prevention systems [5]. In addition, several IDSs and IDS architectures that exploit distributed resources have been proposed [6-8].

Often, a security system only protects its intranet. Network security is similar to that of the real world. A society generally has its own defending policies to protect itself from outside attacks. Many crimes are then exposed. A part of them is dealt with through regional cooperation; network security can be maintained in the same manner. In fact, the Internet comprises many autonomous network management units [1,9,10]. Intranets and campus networks are typical examples. Furthermore, attacks are issued directly or indirectly at, may be, the same or different locations/units. If we can unite adjacent or nearby network management units to form a united defense infrastructure to protect themselves against attacks cooperatively, the Internet will soon become more secure than before.

DoS (denial of service) and DDoS (distributed denial of service) attacks are notorious for

causing tremendous destructions. Some popular websites, such as Yahoo, Amazon, CNN and eBay, were attacked by them. These attacks can be conveniently issued by attacking tools [11] which, having friendly interfaces, are available on the Internet, and can be easily employed to produce huge and legitimate network traffic of same or different protocols to flood victims. A distributed reflective denial of service (DRDoS) attack is an extension of the DDoS by deploying hosts as reflectors. SYN_ACK flooding packets responding from reflectors through HTTP (web) port 80 or some other frequently used ports are hard to distinguish from those of normal connections. Recent research has shown that more than 90% of DDoS and DRDoS attacks are TCP-based [11,13] since three-way handshake lacks an authentication mechanism.

Generally, IDSs based on detective features can be classified into network-based, host-based and a combination of the two, among which the majority are behavior-based. Most traditional network-based IDSs (NIDSs) detect anomalies by monitoring whether network traffic exceeds its threshold or not. Such systems are no longer appropriate or sufficient nowadays due to the diversity of Internet activities. Besides, if we can trace back to the hackers after or during an attack, it would be able to deter hackers from performing their malicious behaviors.

In this article, we propose a security system, named the Intrusion Detection, Forecast and Traceback System (IDeFT), that can effectively detect and forecast intrusions and trace intrusion paths and hackers based on a united defense infrastructure. Each network management unit installs an IDeFT as its security system.

An attack may be a logical one or DoS/DDoS. The former exploits vulnerabilities or security loopholes of a target system. IDeFT detects this type of attack by neural networks [1]. It gathers packet statistics and invokes the Cumulative SUM (CUSUM) [11,14] as its detection algorithm to detects flooding-based DDoS at node level instead of at unit level. Hence, the role a node plays in such an attack can be identified, consequently saving most of the efforts required by IP traceback. However, in some situations, detector can not find underlying attacks, e.g., a DDoS issued by many attackers so that the traffic at each node does not reach its threshold. A hash-based IP traceback mechanism is then employed, no matter whether source IPs of attack packets are spoofed or not. Many traceback systems can only trace back to hacker's default router. There is no way to know which user (host) issues the packets. The reason is that if the spoofed source IP does not belong to any subnets of a router, the router will not record the packet in its ARP (address resolution protocol) table. In case the spoofed source IP truly belongs to one of its subnets, we can realize who submits the packet from the MAC recorded in ARP table. In this study, we assume that MAC can not be spoofed.

To proactively forecast intrusions for an important downstream network management unit, IDeFT investigates transferred (forwarded) flow to locate malicious behaviors launched to the downstream unit, which can then respond properly in advance to minimize damage. Victims can select tracing hackers or protect themselves by filtering out spiteful packets. Through coordination and collaboration of IDeFTs/network management units, we can ensure a more secure Internet. In the following, we use hacker, attacker and intruder interchangeably, even though some consider them to be different terms.

The rest of this article is organized as follows. Section 2 describes related research of this study. Section 3 details our IDeFT system architecture and major components. Section 4 presents the design and implementations of our intrusion detection subsystem. Section 5 defines our overall system defending policies and shows experimental results. Section 6 states conclusions and future research.

II. Related Work

2.1. DDoS Attacks

DoS/DDoS once initiated will continue until it is terminated by hackers or mitigated by victims. Situation becomes worse when control messages of attack packets are encrypted, such as by using CAST-256 algorithm (RFC 2612), to evade IDS's detection.

Establishing a TCP session requires the TCP three-way handshake**,** whereas disconnecting a TCP connection normally needs the exchange of four packets [15]. After accepting a connection request (without verification) from a client, the server allocates resources for the request and sends a SYN_ACK back. This is a so-called half-open connection which will be held until the server receives an ACK or RST from the client. A traditional SYN flood delivers a large amount of SYN packets, all with randomized source IPs, to consume server's resources or target network's bandwidth in order to prohibit server from providing services to legal users [1]. However, servers are often crashed by these connections.

2.2. Distributed Reflective Denial of Service (DRDoS) Attacks

Paxson [16] analyzed reflector attacks in early 2001 and noted that any IP host that returns a reply indicating it has received a packet may act as a reflector. Using traceback techniques, we can trace to reflectors, but can not identify who issued the attack. Huang et al. [17] developed a stepping-stone detection algorithm that analyzes correlations between outgoing streams of outgoing connections and outgoing streams of incoming connections. However, the algorithm only works well for two special scenarios, not generally effective.

2.3. Algorithms and Strategies for Protection

Many countermeasures have been proposed to defend against DDoS attacks [18,19]. Some prevent hosts from becoming zombies. Some try to minimize attack damage. They can be generally divided into three types [18], including

(1) Attack Prevention and Preemption: Security systems of this type prevent DoS/DDoS attacks from taking place. They detect attacks by using known signatures and finding scanning procedures. Some of them further monitor network traffic for known attack messages or spy on attack plan created by, for example, a group of agents [20].

(2) Attack Detection and Filtering: Security of this type is to set up NIDS and/or host-based IDS (HIDS) to detect and filter malicious packets [20,21]. Carl and Kesidis [22] classified detection approaches into activity profiling, sequential change-point detection and wavelet analysis. Chang [18] compared four detect-and-filter approaches on the sets of criteria: detection and filtering mechanisms, effectiveness of attack detection and packet filtering and other technical requirements and deployment effort. Due to spoofed IPs, filtering packets may also impede legal users from accessing network services. Hence, the effectiveness is limited and this approach is sometimes infeasible.

(3) Traceback and Identification: Security of this type is to locate the stream source of spoofed packets so as to identify who the hackers are. However, most traceback techniques require router supports [23,24]. This may heavily increase router load and unfeasibly modify router functions. Most traceback mechanisms mainly focus on how to identify traffic sources [23-28]. Gao and Ansari [29] further analyzed the advantages and disadvantages of traceback schemes and classified these schemes with five selected aspects, including basic principle, processing mode, functionality supported, location and requirement for extra infrastructure. Authors also compared the performance of two stochastic approaches: PPM and iTrace. Jin and Yang [28] compared the performance of DDP and DDP-RD. Belenky and Ansari [26] divided current traceback schemes into four types, including end-host storage [23,24], packet logging [25], specialized routing [30,31] and state of network interference [32].

2.4. Intrusion Detection Systems

Being notorious for high false alarm rate, most IDSs adopt passive policies, i.e., passing malicious packets and alerting administrator for artificial processing rather than dropping them or resetting connections real-time to avoid influencing normal services due to misjudgment. This regrettably results in losing on-line property, even though IDSs work on-line.

The detecting algorithms can be classified into anomaly detection [33,34], rule-based detection [35] and hybrid approaches [36]. The former analyzes audit-logs to identify abnormal users and system behaviors, whereas rule-based systems monitor system logs and resources to uncover malicious packets or attack models that match attack patterns. Anomaly detection, requiring no knowledge base of attack features, learns how a user and system usually behave. However, inaccurate and/or incomplete profiling can lead to false alarms. Therefore, profiles must be frequently maintained.

Rule-based IDSs are unable to detect new attacks that might be recognized by an anomaly detection system. Hybrid IDSs combine the advantages of several types of techniques, while eliminating some of their disadvantages.

Luo et al. [37] proposed two improved one-class anomaly detection models using Support Vector Data Description based on the Cost-Sensitive Learning method. They claimed that their two new models have a low false positive rate compared with a traditional model. Its true positives increase by 22 % and 23 %, while false positives decrease by 58 5 and 94 %.

Other intrusion detection techniques, e.g., forensic approaches and data mining methods, can be found in [38-43].

III. System Framework

IDeFT, as shown in Fig. 1, consists of a sentinel, response manager detector, and tracer. Its working environment is illustrated in Fig. 2.

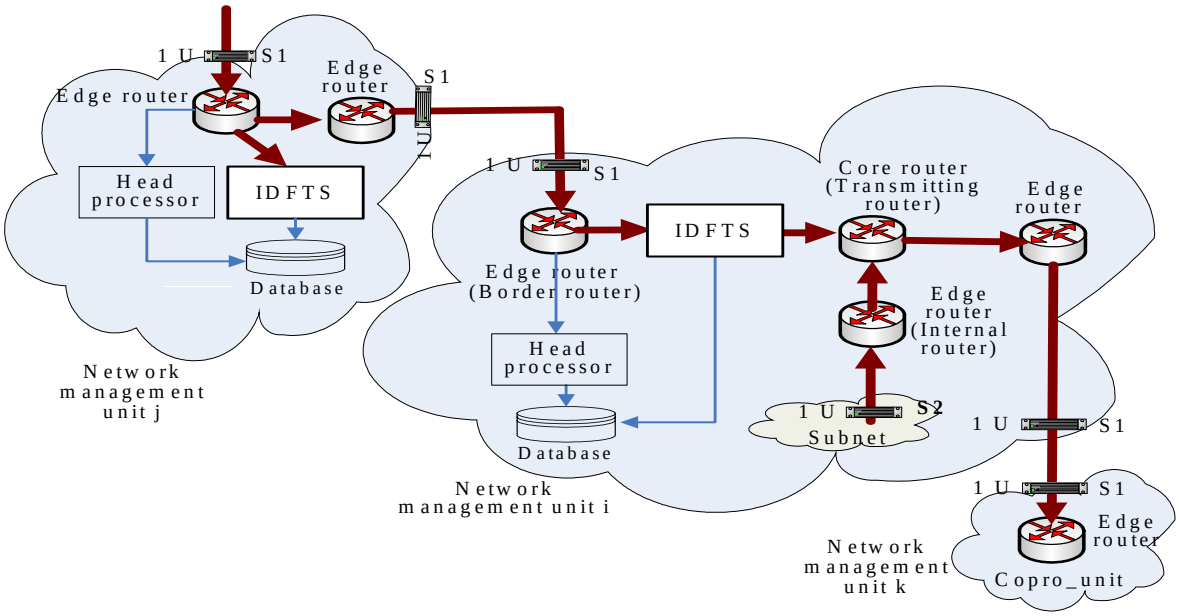

Fig. 1 The IDeFT framework

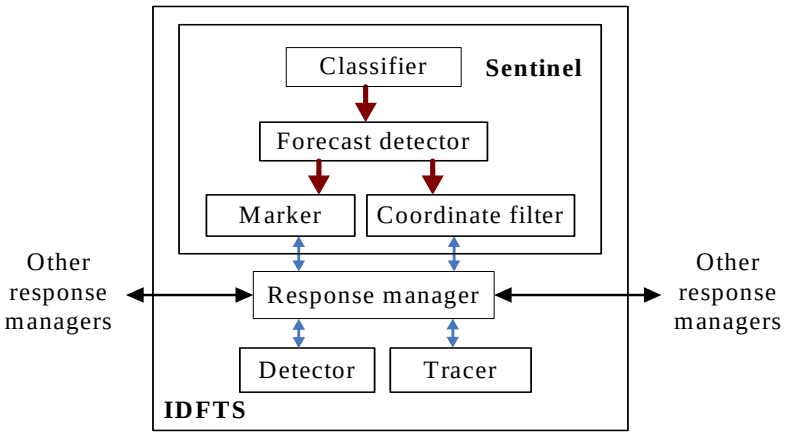

Fig. 2 The IDeFT working environment

Packets that flow through a router according to their destination can be classified into ingress, egress, and forwarded. Detector monitors ingress and egress packets to detect malicious behaviors. Tracer, a log-based traceback subsystem, is responsible for tracing intruders. Response manager, the coordination and administration center of IDeFT, manages detection and trace back activities. It communicates with other network management units to achieve collaboration. Sentinel pre-detects forwarded traffic to discover attacks launched to the network management unit collaboratively protected, named copro_unit.

Often, forwarded packets are detected by their destination network management unit, since they do not threaten the underlying unit, but do increase this unit's burden if detection is performed there. However, if this unit has enough capability to handle extra work, it of course can assist the destination unit to detect intrusions in advance, i.e., perform intrusion pre-detection.

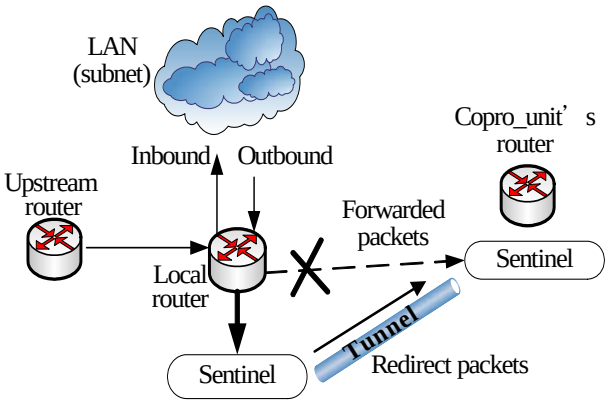

Fig. 3 Redirect the forwarded packets to sentinel

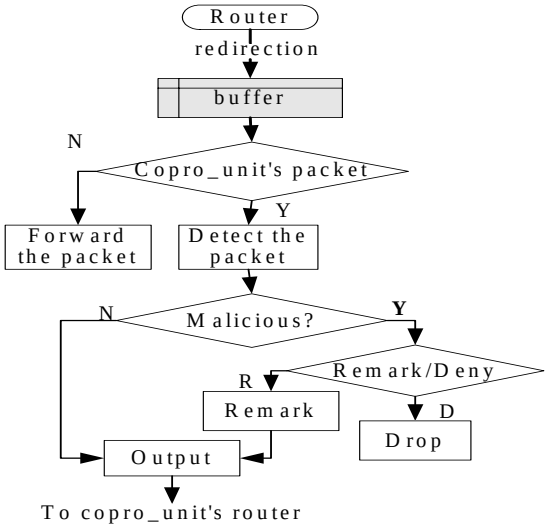

Fig. 4 The sentinel's flow chart

3.1. Forecast Model

When flowing through a router, ingress and egress packets are delivered normally. Forwarded packets are redirected to sentinel, as shown in Fig. 3. When sentinel discovers an attack, response manager sends an alerting message, which can be in-band or out-band, to response manager of the copro_unit. In-band messages are short messages inscribed on underlying malicious packets, whereas out-band messages are carried by independent packets, thus resulting in extra network burden. Besides, an out-band message may arrive at its destination (the victim) after the victim is damaged by the malicious packet. A forecast model should at least follow the following guiding principles:

(1) Forecast is not performed completely. In a copro_unit, only hosts or servers playing key roles in service and control are collaboratively protected. In IDeFT, their IPs are collected in a protecting list.

(2) Most attacks can be detected in upstream nodes, which however should not consume too much bandwidth and too many resources. Therefore, only destructive or ruinous intrusions are pre-detected.

(3) An alerting message and its communication channel should be trustworthy, i.e., not be destroyed or faked.

(4) The communication process should be protected, especially from any connection hijacking.

3.1.1. Sentinel

Sentinel, acting like a remote IDS and/or firewall, mainly consists of a classifier, forecast detector, marker and coordinate filter, as shown in Fig. 1. Classifier classifies packets into two classes, in protecting list and out of protecting list. Only in-protecting-list packets of TCP, ICMP and UDP protocols are enqueued in buffer, in which malicious ones are audited by forecast detector for further processing based on pre-defined policies, e.g., dropping by coordinate filter or marking an alerting message by marker. Coordinate filter filters packets according to the policies. Except when dropped, all packets including normal and marked are all sent to their original destination network management units. Fig. 4 shows the sentinel's flow chart.

3.1.2. Alerting Message

In an IP header, identification (ID) field and its following reserved bit, a total of seventeen bits named marked field as shown in Fig. 5, is deployed to convey an in-band alerting message [44,45]. ID field was originally defined to identify fragments of a datagram. Song and Perrig [27] pointed out that only 0.25% packets take this field for use in network transmission, i.e., a low-entropy field. Very few packets, only malicious and suspected ones, are marked by our scheme. The influence on normal service is limited. The probability is 0.0025p1p2, where p1 is the probability of false positive, i.e., the probability that a normal packet is mis-judged as an attack packet, and p2 is the probability that in-band alerting is chosen.

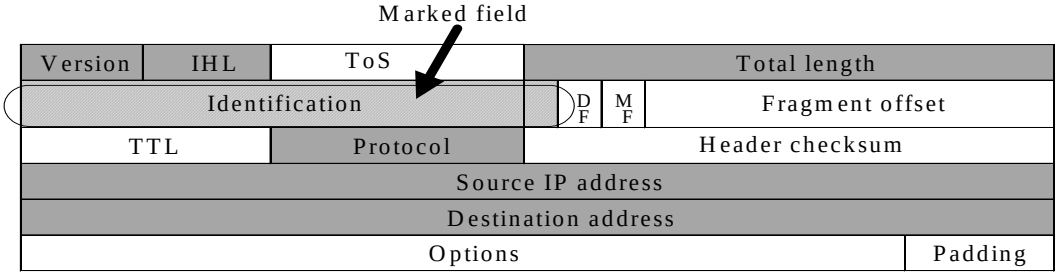

Fig. 5 Marked field (Identification field plus its following bit) and unchanged (shaded) fields in IP header during transmission

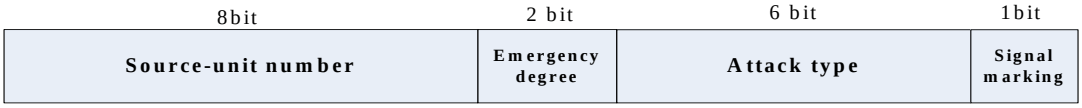

Fig. 6 Format of an alerting message (a total of 17 bits)

An alerting message, as illustrated in Fig. 6, consists of four fields, including 8-bit source-unit number, 2-bit emergency degree, 6-bit attack type and 1-bit signal marking fields. Source-unit number shows which network management unit marks underlying alerting message. Therefore, up to 256 units are allowed to cooperate with each other as a group. Emergency degree represents dangerous degree of a packet:

(1) Emergency degree =0: Having malicious intent or characteristics, such as scan port
(2) Emergency degree =1: An attack packet
(3) Emergency degree =2: A suspected DoS/DDoS
(4) Emergency degree =3: DoS/DDoS attack

Attack type indicates the attack categories, e.g., SQL slammer DDoS or bandwidth consumption DDoS. Emergency degree and attack type together can present a total of 2^8 types. Normally, an in-band alerting message is chosen. The following is the marking algorithm:

Algorithm: Marking an in-band altering message

Input: packet P

Output: P is encoded with an alerting message if it is a malicious packet

{ if P is malicious then
{ mark signal marking field; /* indicating P is malicious */
encode underlying network management unit ID to source-unit number field;
encode emergency degree to emergency degree field;
encode the attack type detected by forecast detector into attack type field**;**
send P to its original destination. }}

Moreover, some attack types, e.g., a variation of a known DDoS, may not completely match existing signatures due to evolution of intrusion techniques. Since we are not sure, only suspect, that may be there is an attack. No packet ought to be dropped or marked at this point. An out-band alerting message is then chosen. The receiver replies to the sender also with an independent packet.

3.2. Response Manager

Response manager, as shown in Fig. 7, comprises a coordinator, policy database, certification authority, register authority and query agent. When detector finds an illegal behavior, it notifies coordinator to respond appropriately based on policies predefined in database, such as discarding the packet, terminating the session, reconstructing the datagram, raising the emergency degree or tracing the attack source. Generally, *Trace* *(from_response manager)*, a request sent to its neighbor response manager for tracing, is issued by coordinator.

In addition, certification authority and register authority [18] are deployed to prevent reception of faked messages and to avoid being hijacked, i.e., making IDeFT satisfy the third and fourth guiding principles of a forecast model. Certification authority authenticates the identities of response managers a response manager communicates with. Register authority is responsible for identity registration and deregistration. To join this defense organization, a network management unit must register itself to local register authority with the information, including network management unit ID, response manager's IP, MAC addresses of all edge routers and protecting list if this network management unit needs to protect copro_units. Register authority gives the unit's response manager a certificate and sends relevant information to certification authority, which then returns a signature to the response manager.

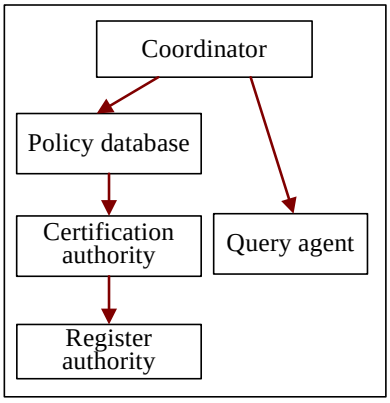

Fig. 7 The response manager's Framework

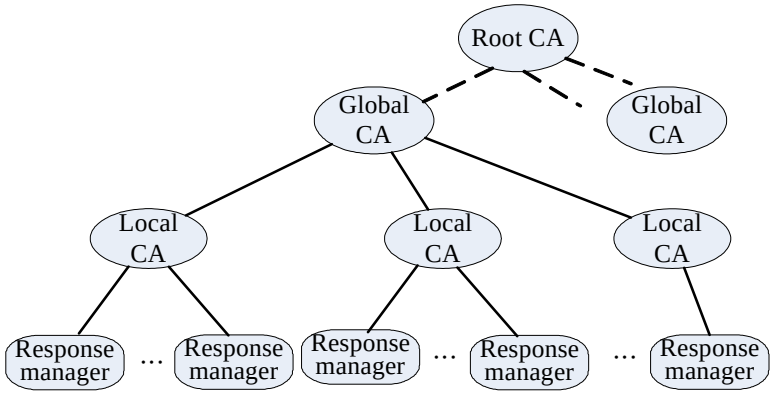

Fig. 8 The hierarchy scheme of response managers (CA: certification authority)

The relationship among certification authorities as shown in Fig. 8 is a hierarchical structure. Nearby certification authorities are grouped as the child nodes of a local certification authority. Several local certification authorities form a group which is rooted by their parent certification authority, i.e., a global certification authority. A parent certification authority duplicates all the information from its child certification authorities.

Before communicating with a remote response manager, a local response manager invokes query agent to query the remote response manager's information. The remote manager then authenticates the local one's identity through certification authority and returns queried information to the local manager.

3.3. Tracer

3.3.1. Tracing Attack Sources

Tracer uses "electronic watermarks", also named digests, as the tracing basis. Several switches that provide mirror ports are deployed in a network management unit. They can be classified into type 1 and type 2. A switch of type 1, e.g., S1 shown in Fig. 2, is placed on the link connecting an edge (a border) router to another network management unit. A type 2 switch, e.g., S2, is located on the link connecting an edge (internal) router to one of its subnets. When a packet flows through S2 from the subnet, S2 collects the packet's source IP, source MAC and current time to establish an IP-MAC mapping table for future trace. All IP-layer packets flowing through S1 are captured by a sniffing AP (such as tcpdump). A software module named header processor for collecting and processing packets retrieves the necessary information from each packet, including 40-byte packet data and the unchanged fields during packet transmission, such as source and destination IPs (shown as shaded areas in Fig. 5), but excluding marked field, and encodes the information with hash functions, such as MD5, to generate digests as packet identifiers.

The 40-byte data not only reduces digest collision, but also supports detector for auditing since some attack packets carry the same data, which is clearly a detection pattern/feature. Several hash functions are used to further reduce collision. The hashed results are stored in the bloom filter [25]. The data before and after hashing are 54 bytes and 16 bits in length, respectively. The latter together with source MAC and timestamp form a trace record which is used for tracing hackers.

Generally, if a network management unit has N adjacent neighbors, in the worst case, N network management units need to be traced before we can identify the upstream one. Hence, tracing to the intruder, we have to trace a total of $N*i$ network management units if the intruder's unit is i units away from us. However, a source MAC address in a trace record can reduce tracing cost from $N*i$ to i.

Most router-based tracing mechanisms increase router load [23,24]. However, during a DoS/ DDoS attack, routers are too busy to handle extra work. Digests and traceback in IDeFT are respectively generated and performed by additional devices. This can minimize influence on the router's performance. Moreover, with digest, tracing hackers becomes lightweight and consumes less bandwidth.

3.3.2. Tracing Procedure

When detector i discovers that packet P is malicious, or receives an alerting message from an upstream network management unit, it sends a request message $Trace\_request_{(from\_detector)}$, which consists of P's source MAC (e.g., MAC_P), P's digest, P's source IP (e.g., IP_P) timestamp and IP address of the victim's response manager (e.g., IP_V) to notify tracer i (the tracer of network management unit i). Timestamp can reduce the searching space for each unit [25]. Table 1 lists the messages deployed in IDeFT. The following is our tracing procedure from a global perspective.

(1) After receiving a tracing request, tracer i asks query agent to check which network management unit MAC_P belongs to (assumes network management unit j) to obtain response manager j's IP address, and then reports to response manager i. /* performed by network management unit i*/

(2) Response manager i sends a tracing message $Trace_{(from\_\ response\ manager)}$, which consists of P's digest, IP_V and IP_P, to response manager j;

(3) Response manager j queries its database to check for the existence of *P's* digest. /* performed by network management unit j*/

(4) If not (i.e., a faked MAC_p), response manager j sends a *Trace_result* to the trace-initiating response manager to report the exception; stop.

(5) If yes, response manager j retrieves the corresponding source MAC (e.g., MAC_k) from the underlying record to check whether the next node should be traced or not.

(5.1) If yes (i.e., not local MAC), response manager j accesses response manager k's IP according to MAC_k; let i=j and j=k; go to step (2).

(5.2) If not, response manager j

(5.2.1) searches network management unit j's IP-MAC mapping table based on IP_P to retrieve the corresponding MAC (e.g., MAC_A);

(5.2.2) notifies the local administrator to deal with the intrusion;

(5.2.3) reports to the trace-initiating response manager with a message *Trace_result*, which comprises IP_j and a result description containing IP_P, digest and MAC_A.

Table 1 Message types

Message Types	*Message Content*	*Description*	*Types*
Messages for tracing			
Trace_request $_{(from\_\ detector)}$	**MAC_P, digest, IP_P, timestamp, IP_V**	Initiate tracing	Out-band
Trace $_{(from\_\ response\ manager)}$	**Digest, IP_V, timestamp, IP_P**	Query between response managers	Out-band
Trace_result	**IP_j , result description**	Reply to trace-initiating response manager	Out-band
Messages for forecasting			
Alerting_message	**Source-unit number, emergency degree, attack type, signal marking**	Forecast attack	In-band
Alerting _message	**Source-unit number, emergency degree, attack type**	Forecast attack	Out-band
Request_for_help	**Response manager's IP, attack type**	Reply to a forecast	Out-band
Request_for_filtering	***IP_V*, attack type**	Ask for filtering attack packets	Out-band
Request_for_marking	***IP_V*, attack type**	Ask for marking attack packets	Out-band

IV. Detector

Detector consists of the CUSUM intrusion detection system (CIDS) and logical attack detector. The latter is discussed in our previous research [1,19,46]. CIDS, an IDS with HIDS and NIDS properties, detects DDoS and DRDoS by collecting all ingress and egress packets as the observed event sequences for the underlying network management unit. A procedure, named change-point detection designed to detect the change point of network behavior [11], is as follows. First, like most statistical anomaly detection systems, CIDS compares the observed sequence with users' normal behaviors recorded in profiles to locate any significant discrepancy or difference. If any is found, it identifies the time point at which the change begins. Second, the CUSUM is deployed to monitor input random variables sequentially. Simple parametric approaches [33,34] are often too simple to model a network session accurately due to the complexity of the Internet. The CUSUM, with the characteristics of a sequential and non-parametric light computation load, can make CIDS work online.

4.1. The CUSUM

The CUSUM is able to detect something that sharply but continuously increases. Some of its assumptions are given below**.** First, let X_n be the packets collected by CIDS within a sampling time Δn and $\overline{X}$ the mean value of random sequence X , $X = \{X_n, n = 0,1,2...\}$. Second, let $Z = \{Z_n\ , n = 0,1,2...\}$ with *a*, where $Z_n = X_n - a$ and a is the peak value of

normal traffic. Hence, all elements of Z are negative or zero that makes $\overline{Z}$ become negative.

When a change, such as a flooding-based attack, occurs, Z_n will suddenly become positive, as illustrated in Fig.9. $Z_k \geq \overline{Z} + h$ indicates the moment an attack may be starting, where k is the smallest n and h the threshold of abnormal network traffic. Δ_k is then considered to be the change point. $y_{n-1} + Z_n \leq 0$ shows there is no attack. The CUSUM accumulates Z_n, $n \geq k$, with formula (1), which is the recursive version of the non-parametric CUSUM algorithm [11].

$$y_n = (y_{n-1} + Z_n)^+, y_0 = 0 \quad (1)$$

where $x^+ = x$ if $x > 0$ and 0 otherwise. $Z_n, n > k$, may now be positive or negative. The decision function at Δp, e.g., $d_p(y_p)$, is as follows.

$$d_p(y_p) = \begin{cases} 1 & \text{if } y_p > N \\ 0 & \text{else} \end{cases} \quad (2)$$

where N is attack threshold and '1' indicates there is an attack.

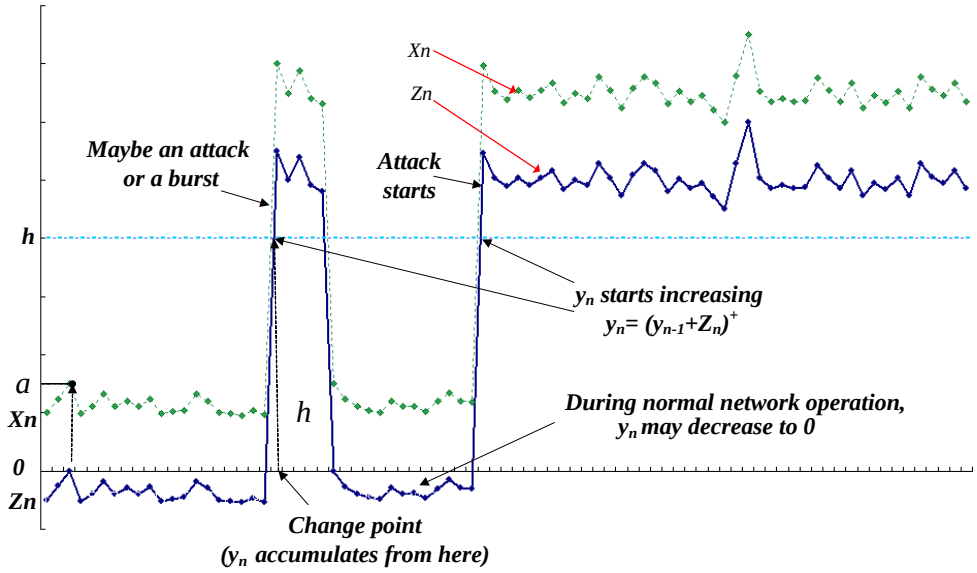

Fig. 9 The behavior of the CUSUM

4.2. TCP Flood

In the following, we analyze traditional and reflective TCP flood attacks in detail. A host inside a network management unit may act as a victim, reflector, or an attacker (zombie).

(1)Traditional SYN Flood

During an attack, as shown in Fig. 10a, because SYN_ACK packets never be replied back to zombie, $|O-SYN|$ and $|I-SYN\_ACK|$ at zombie significantly differ, where $|X|$ represents the number of X, $O-SYN$ and $I-SYN\_ACK$ stand for outgoing SYN and incoming SYN_ACK, respectively. Basically, $|I-SYN|$ and $|O-SYN\_ACK|$ at victim are equally huge.

Generally, an RST packet following a TCP packet (e.g., SYN, SYN_ACK, URG or PSH) may result from one of three cases: (1) terminating a TCP session; (2) aborting a TCP request; or (3) sending a packet to a closed port or an unconnected node. The first two cases are strongly related to SYN.

Normally, no matter inbound or outbound traffic, |SYN|+|SYN_ACK| at a node is roughly equal to its $|RST|+|FIN|$. The number of active RST packets generated by the first two cases is about 75% of all RST packets. The remaining 25% resulted from the third can be treated as background noise. This occurrence will improve detection sensitivity and decrease false alarms [15].

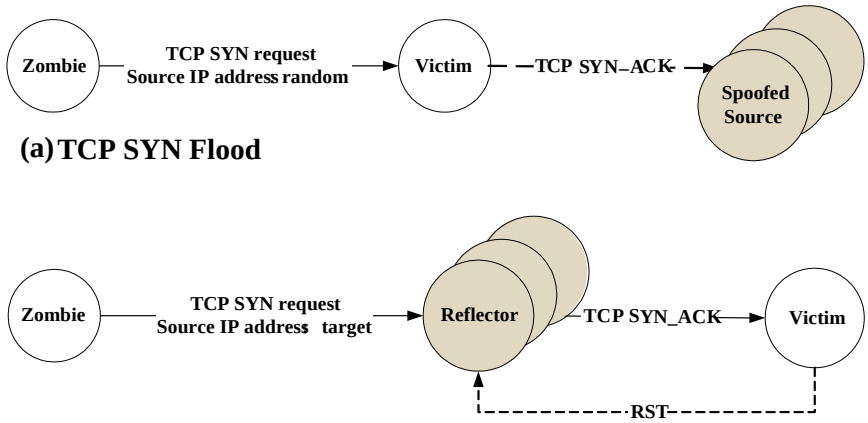

Fig. 10 Attack nodes of TCP flooding

CIDS treats (SYN, FIN), (SYN_ACK, FIN), and (SYN, RST_{active}) pairs occurred at victim as normal behaviors. A deviation indicates the node may be under a TCP SYN flood attack.

(2) Reflective TCP Floods

During a reflective attack, zombies send some level of volume of TCP SYN requests to each reflector [47,48]. Two facts are useful in detecting such an attack. First, when receiving a spoofed request, a reflector replies with a SYN_ACK to the victim which, as shown in Fig. 10b, will send an RST "back". Second, when any TCP packet other than SYN, such as URG, PSH, FIN or a TCP packet without a flag, arrives without any previous handshake, the reflector replies with an RST to the victim, which as illustrated in Fig. 10c gives no answer.

From the second fact mentioned above, we can conclude that during a TCP SYN flood $|I-SYN\_ACK|$ and $|O-RST|$ at victim and $|O-SYN\_ACK|$ and $|I-RST|$ at reflector all hugely increase. But $|O-SYN|$ and traffic generated by other TCP packets at both nodes do not. These phenomena are extremely useful in detecting reflectors and victims.

4.3. Detecting TCP Types of Floods

CIDS detects TCP flood by calculating corresponding X_n.

(1) **Victim**

Let $|S_{SYN}|$, $|S_{FIN}|$, $|S_{SYN\_ACK}|$ and $|S_{RST}|$, respectively, be numbers of SYN, FIN, SYN_ACK and RST packets observed within a sampling interval Δn, but ignoring packet direction. Let

$$X_{Vn} = \frac{|S_{SYN}| + |S_{SYN\_ACK}| - (|S_{FIN}| + |S_{RST}|)}{(|S_{FIN}| + |S_{RST}|)} \quad (3)$$

which is independent of network size and ordinary traffic since normally the ratio between SYN and SYN_ACK and their reply pairs (RST and FIN) is relatively stable. Under normal network condition, few RST packets are generated. The main reasons that $|S_{SYN}| + |S_{SYN\_ACK}|$ significantly differs from $(|S_{FIN}| + |S_{RST}|)$ are described in [11]. With X_{Vn}, a victim can be discovered**.** However, Thompson et al. [49] mentioned that most TCP connections last 12-19 seconds. Let $\Delta n = 10\,\text{sec}$, FIN and RST will always appear in the next or next second sampling interval of the one where the corresponding SYN locates. Let the sampling time correction (delay) of $|S_{FIN}|$ and $|S_{RST}|$ be β, i.e.,

$$\Delta n_{(FIN,\,RST)} = \Delta n_{SYN} + \beta$$

Here, $\Delta n_{(FIN,RST)}$ and Δn_{SYN} are the sampling intervals of S_{FIN} (or S_{RST}) and S_{SYN}, respectively. Wang et al. [15] supported the way to quantify parameters, and proposed that $\beta = 10$ could balance detection sensitivity and accuracy. CIDS adopts them.

Given parameters of X_n, i.e., (h_v, a_V, N_V), the corresponding decision function can determine if a node *V* is a victim of a traditional TCP SYN flood or not. However**,** intruders may maliciously send FIN and RST packets to compromise detection accuracy. To solve this problem, we need a supporting parameter X_D, which will be described later.

Let $$X_{Vn}' = \frac{|I - SYN\_ACK| + |O - RST| - |O - SYN|}{|O - SYN|} \quad (4)$$

Similarly**,** given (h_v', a_v', N_V'), CIDT can decide if the node concerned is now a victim of a reflective TCP SYN flood or not. This can be very accurate since an intruder has no way to generate spoofed O-SYN to decrease X_{Vn}'**.**

Let $$X_{Vn}'' = \frac{|I - RST| - avg|I - RST|}{avg|I - RST|} \quad (5)$$

where *avg* represents average. We can detect victims of a reflective TCP flood with X_{Vn}''.

(2) Attacker (Zombie)

On the attacker side, let

$$X_{Zn} = \frac{|O - SYN| - |I - SYN\_ACK|}{|I - SYN\_ACK|} \quad (6)$$

Given (h_Z, a_Z, N_Z), the CUSUM can determine if a node is a zombie or not for both traditional and reflective TCP SYN floods. The attacker shown in Fig. 10c can be detected by counting the difference between the number of total output and input packets.

Let $$X_{Zn}' = \frac{|O - X| - |I - X|}{|I - X|} \quad (7)$$

where X is any type of packet other than SYN.

(3) Reflector

Let $$X_{Rn} = \frac{|I - RST| - avg|I - RST|}{avg|I - RST|} \quad (8)$$

Given (h_R, a_R, N_R), reflectors of a reflective TCP SYN flood can be probably detected. However, observing X_{Rn} may be insufficient since its $|I - RST|$ may be not huge enough,

especially when many reflectors are deployed (assuming k reflectors, i.e., $N_R = N_V'/k$). When k>> default value, N_R may result in false negatives. In contrast, when k << the default value, a false positive may occur. To solve this problem, a statistical process or observation to obtain the average value of k is required. Besides, an abnormal increase in both X_{Rn} and X_{Zn} in the same or different network management units indicates there is a reflector. This finding is the basis of starting the identification of reflectors. Reflectors should be those connecting to zombie and victim at the same time. Also, when reflectors and zombies are located in different network management units, their response managers have to communicate with each other.

Let $$X_{Rn}' = \frac{|O-RST|+|I-X|-avg|O-RST|}{avg|O-RST|} \quad (9)$$

where X is any type of packet other than SYN. Given (h_R', a_R', N_R'), the reflector of the reflective TCP flood can be detected. Besides, another supporting parameter X_D' is required to discriminate reflectors of Fig. 10b and victims of Fig. 10c due to $X_{Rn} = X_{Vn}''$. Let

$$X_D' = \frac{|S_{RST}|+|S_{FIN}|}{|S_{SYN}|+|S_{SYN\_ACK}|} \quad (10)$$

X_D' is stable under normal network operation. During a reflective TCP SYN attack, $|S_{SYN}|$, $|S_{SYN\_ACK}|$ and $|S_{RST}|$ increase, but $|S_{FIN}|$ does not, and $|I-RST|$ is almost equal to $|O-SYN\_ACK|$. Thus X_D' decreases. But in Fig. 10c, only the $|I-RST|$ becomes huge resulting in an abnormal increase in X_D'.

During a TCP SYN flood, as stated above, an attacker may send a huge amount of FINs and/or RSTs to decrease X_{Vn}. The supporting parameter X_D (=$|S_{RST}|+|S_{FIN}|$) calculated at victim will significantly increase since the amounts of compensating packets required and attack SYNs are roughly equal, i.e., X_{Rn}' and/or X_{Vn}'' becomes huge. Our conclusion is that once a node is detected either as a reflector of a reflective TCP SYN flood or a victim of a reflective TCP flood, and its X_{Vn} is still normal, the attack is clear a hybrid, mixing a traditional TCP SYN flood with either a reflective TCP SYN flood or a reflective TCP flood. According to our observation, 99% are reflective TCP SYN floods since it is not easy to generate FIN packets. Table 2 summarizes the detection approaches.

Table 2 Summary of detecting approaches (↑ :value goes up; ↓ :value goes down)

Attack Type	**SYN flood (DDoS)**			**Reflective SYN flood (DRDoS)/ Reflective TCP flood**					
Network behavior	O- SYN	I-SYN	I-SYN_ ACK	O- SYN	I- SYN	I-SYN_ACK	O-SYN_ACK	O- RST	I-RST
Victim	Normal	**Increase**	Normal	Normal/ normal	Normal/ normal	**Increase**/ Normal	Normal/ normal	**Increase** / normal	Normal/ **increase**
Reflector	-	-	-	Normal/ normal	**Increase**/ normal	Normal/may increase	**Increase**/ normal	Normal/ **increase**	**Increase**/ normal
Attacker	**Increase**	Normal	Normal	**Increase** / normal	Normal/ normal	Normal/ Normal	Normal/may increase	Normal/ normal	Normal/ normal
Detecting type (Victim)	Abnormal difference between $\|S_{SYN}\|+\|S_{SYN\_ACK}\|$ and $\|S_{FIN}\|+\|S_{RST}\|$ & supporting parameter $X_D = \|S_{RST}\|+\|S_{FIN}\|$			Abnormal difference between $\|I-SYN\_ACK\|+\|O-RST\|$ and $\|O-SYN\|$, Abnormal difference between $\|I\text{-}RST\|$ and its average value & supporting factor $X_D' = \frac{\|S_{RST}\|+\|S_{FIN}\|}{\|S_{SYN}\|+\|S_{SYN\_ACK}\|}$					
***Xn* in CUSUM (X_{Vn})**	$X_{Vn} = \frac{\|S_{SYN}\|+\|S_{SYN\_ACK}\|-(\|S_{FIN}\|+\|S_{RST}\|)}{(\|S_{FIN}\|+\|S_{RST}\|)}$			$X_{Vn}' = \frac{\|I-SYN\_ACK\|+\|O-RST\|-\|O-SYN\|}{\|O-SYN\|}$, $X_{Vn}'' = \frac{\|I-RST\|-avg\|I-RST\|}{avg\|I-RST\|}$ & $X_D' \uparrow$					
Meaning of a_V in CUSUM	Peak value of X_{Vn} under normal network operation			Peak values of X_{Vn}' and X_{Vn}'' under normal network operation					
Detecting type (Reflector)	None			Abnormal difference between $\|I-RST\|$ and its average value & supporting parameter X_D', Abnormal difference between $\|O-RST\|+\|I-X\|$ and average value of $\|O-RST\|$					
***Xn* in CUSUM (X_{Rn})**	None			$X_{Rn} = \frac{\|I-RST\|-avg\|I-RST\|}{avg\|I-RST\|}$ & $X_D' \downarrow$, $X_{Rn}' = \frac{\|O-RST\|+\|I-X\|-avg\|O-RST\|}{avg\|O-RST\|}$					
Meaning of a_R in CUSUM	None			Peak values of X_{Rn} and X_{Rn}' under normal network operation					
Detecting type (Zombie)	Abnormal difference between $\|O\text{-}SYN\|$ and $\|I\text{-}SYN\_ACK\|$			Abnormal difference between $\|O\text{-}SYN\|$ and $\|I-SYN\_ACK\|$, Abnormal difference between $\|O-X\|$ *and* $\|I-X\|$ where X is any packet other than SYN					
***Xn* in CUSUM X_{Zn}**	$X_{Zn} = \frac{\|O-SYN\|-\|I-SYN\_ACK\|}{\|I-SYN\_ACK\|}$			$X_{Zn} = \frac{\|O-SYN\|-\|I-SYN\_ACK\|}{\|I-SYN\_ACK\|}$, $X_{Zn}' = \frac{\|O-X\|-\|I-X\|}{\|I-X\|}$					
Meaning of a_Z in CUSUM	Peak value of X_{Zn} under normal network operation			Peak values of X_{Zn} and X_{Zn}' under normal network operation					

V. Defending Policies and Experiments

5.1. Defense with Forecast Model

When network management unit i discovers that one of its protected hosts is under DDoS attack, it sends alerting message to corresponding copro_unit (e.g., network management unit p) whose detector, detector p, confirms the attack. Its response manager, response manager p, then replies with the message ***Request_for_help*** to request response manager i to trace intruders which are located, assuming, at network management unit Z (known by back trace). In this study, two defending policies, as shown in Fig. 11, are used:

1. Response manager i sends a message ***Request_for_filtering*** to ask network management unit Z to filter packets sent to the victim to prevent all the nodes (routers) along the attack path from exhausting their bandwidth. Its drawback is it loses relevant attack information.
2. Response manager i sends a message ***Request_for_marking*** to ask network management unit Z to mark alerting message. Coordinate filter i collects and analyzes alerting messages to continue accumulating new features for the improvement of future forecasting efficiency and accuracy. It finally drops these packets to mitigate the victim's damage. The lower portion of Table 1 lists the messages for forecasting.

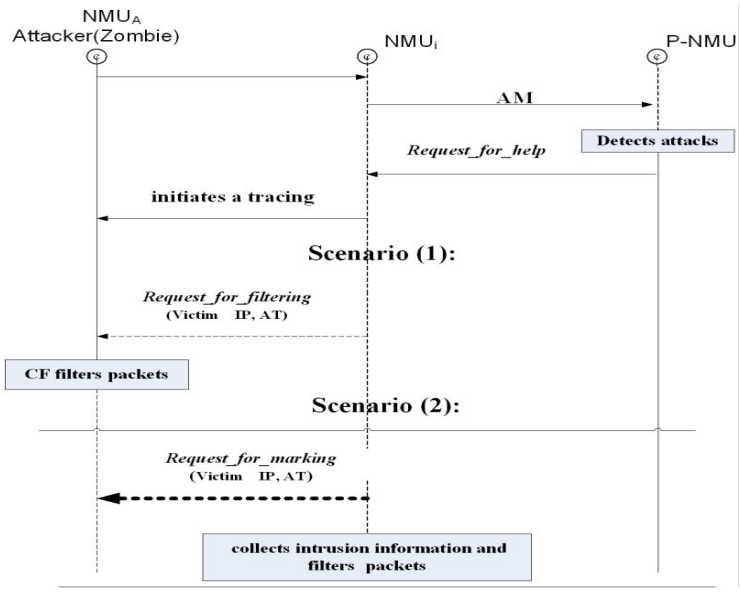

Fig. 11 Two defense scenarios

5.2. Defense with Network Management Unit

Traceback and filtering mechanisms are insufficient to defend against DRDoS since zombies often change reflectors randomly. When finding malicious behavior, CIDS counts inbound packets sent to the attacked subnet (recall, collected by type 1 switch) according to the packets' destination IP and source MAC addresses. Through source MAC, the number of packets issued from an upstream router can be correctly calculated even if their source IP addresses are spoofed.

Once a suspected attacker, reflector or a victim is found, CIDS asks its own response manager to notify the victim's network management unit, which may be local or remote, to respond properly. Our traceback mechanism can trace the intrusion paths from victim to reflectors and from reflectors to zombies under collaboration of network management units. This is helpful in identifying the nodes along the intrusion paths so that network administrators can check and remove the installed Trojans one by one. Fig. 12 illustrates the difference between scenarios with and without IDeFT.

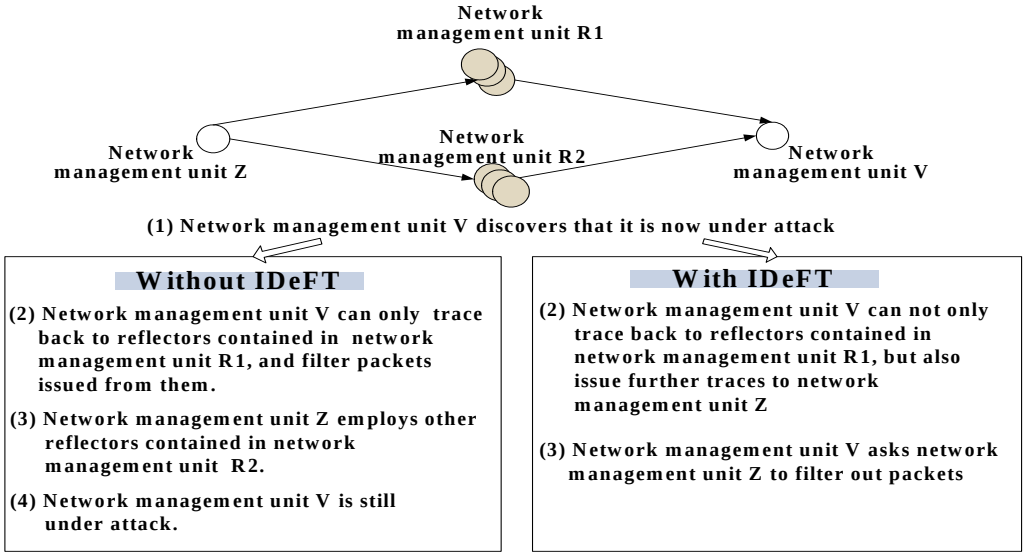

Fig. 12 Difference between scenarios with and without IDeFT

5.3. Experiments

Five experiments were performed. The first was a simulation designed to evaluate tracing efficiency. The second and third measured parameters for zombies, reflectors and victims in a DDoS attack. The fourth and fifth evaluated detection timings and accuracies for different security systems tested, respectively.

(1) Experiment 1: Tracing Efficiency

Given a network, each node is adjacent to $N(=11)$ nodes and the distance from hacker to victim is i(=10). Different numbers of attack packets (100, 500, 1000 …) were issued to measure tracing costs and to count the total numbers of network management units traced when only digest or both digest and MAC were in use. The experimental results are shown in Table 3. We can see that, using both digest and MAC address, the average number of network management units in each trace was 10 ($M2/R1$) rather than about 55 ($M1/R1$). $M1/R1$ is about $(\frac{N}{2})$ times *M2/R1*. The cost is reduced to about one-eighth (see $T2/T1$). Furthermore, tracing a packet (a network management unit) back costs 4.30 (0.43) msec instead of 33.64 (3.364) msec. Table 4 lists efforts that a tested system requires to trace a path or q paths, $q > 1$. Other comparisons can be found in [24, 26, 29].

Table 3 Simulation results of using digest only and both digest and MAC

Total no. of attack packets issued (R1)	Digest only		M1/R1	Digest + MAC		M2 /R1	T2/T1
	Total no. of network manag. units traced (M1)	Cost (ms) (T1)		Total no. of network manag. units traced (M2)	Cost (ms) (T2)		
100	5387	3375	53.9	1000	422	10	1/7.99
500	27723	17281	55.4	5000	2140	10	1/8.07
1000	55389	34594	55.4	10000	4297	10	1/8.05
5000	275245	172141	55.0	50000	21500	10	1/8.00
8000	443201	254500	55.4	80000	34376	10	1/7.40
10000	549740	344047	55.0	100000	43047	10	1/7.99
11000	602802	371501	54.8	110000	47230	10	1/7.86

Table 4 Comparison of different traceback approaches where q attackers are each i steps (network management units) away from victim.

Efforts / System Tested	No. of bits accessed/generated in a packet for tracking (bits)	The least no. of packets addressed /received for tracing one step back (pkts)	The least efforts (in packets) required for reconstructing one path/q paths
ITRACE	**0 (but different information from routers)/x (x>(256+3)*8)**	**2/20000 (marking prob. =1/20000)**	**2i/2qi**
PPM (marking probability *p*)	**0 (but 32 router addr. bits + 32 hash bits)/16 (i.e., 8 fragment bits +5 addr. bits+ 3 offset bits)**	**k(=8)/k**	$\frac{k \ln ki)}{(p(1-p)^{i-1})} \Big/ \frac{qk \ln ki)}{(p(1-p)^{i-1})}$
SPIE	**24*8/2^n (n: size of a digest)**	**1/1**	**1, but hash i times/q, but hash qi times**
DPM	**32/k17 (i.e., |a-bit| (=$\left\lceil \frac{32}{k} \right\rceil$) +digest+seg. no)**	**k/k**	**ki/kqi**
IDeFT (digest)	**54*8/16 (i.e., digest)**	**1/1**	**1, but hash i times/q, but hash qi times**
IDeFT (digest+ MAC)	**(54+6)*8/64 (i.e., digest (16)+MAC (48))**	**1/1**	**1, but hash i times/q, but hash qi times**

(2) Experiments 2 and 3: Parameter Measurements

The environment of the remaining experiments is shown in Table 5. Network segments 140.128.101.*, 140.128.103.*, 140.128.104.* and 140.128.98.* are the observed subnets; each deploys one or several nodes (PCs) as its servers or hosts to simulate zombie, reflectors and victim.

Table 5 Experimental environment

The roles of nodes	IP addresses	No. of nodes employed
Attacker	140.128.98.*	1
Reflectors-subnet 1	140.128.98.*	12
Reflectors-subnet 2	140.128.103.*	8
Reflectors-subnet 3	140.128.101.*	10
Victim's subnet	140.128.104.*	1

Experiments 2 and 3 show the feasibility of CIDS. Let the observed interval $\Delta n_{(FIN,RST)} = \Delta n_{(SYN,SYN\_ACK)} = \beta = 10$ sec. Also, normally the $|O - SYN|$ at a node is always larger than $|I - SYN\_ACK|$ due to the reasons stated in [15] which treated the difference as white noise. However, this phenomenon shown in Fig. 13 has less influence on X_{Zn}, but results in negative $\overline{X_{Vn}}$' under normal network operation. Let $Z_{Vn}' = X_{Vn}'$, i.e., a=0. Besides, according to our observation on X_{Vn} shown in Fig. 14, let $h_V = 4$, $a_V = 2.5$ *and* $N_V = 4$.

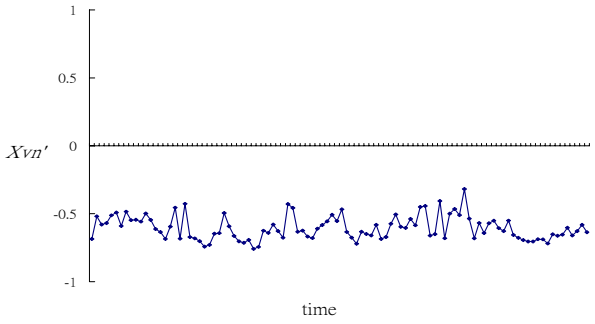

Fig. 13 X_{Vn}' with negative $\overline{X_{Vn}}$' (Reflective SYN flood)

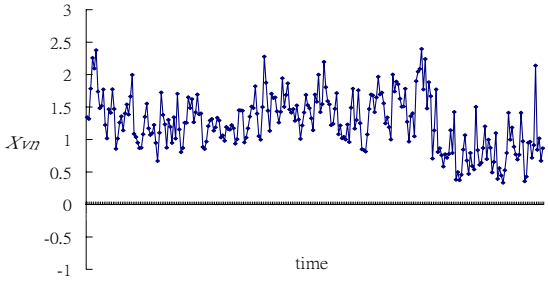

Fig. 14 Example of X_{Vn} (SYN flood)

Experiment 2 was a simulation of DDoS attack. Victim and zombie are located in different network management units, e.g., unit V and unit Z, respectively. We can see that, during an attack, y_{Vn} shown in Fig. 15 sharply increased, but y_{Zn} of each node in unit V did not. In unit Z, some node's y_{Zn} exceeded N_z, as illustrated in Fig. 16, indicating this node is an attacker.

Experiment 3 was a simulation of reflective TCP SYN flood. Some X_{Rn} in network management unit R hugely increased and y_{Rn} soon exceeded N_R, as shown in Fig. 17, indicating it is a reflector. Fig. 18 illustrates that y_{Vn}' in network management unit V also exceeded its threshold, showing that the victim is under attack. The results of reflective TCP flood attacks are similar to those illustrated in Figs. 17 and 18.

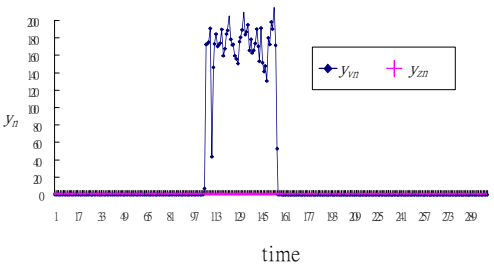

Fig. 15 y_{Vn} and y_{Zn} detected at network management unit V under DDoS attack (victim in unit V and zombie in unit Z, $v \neq z$)

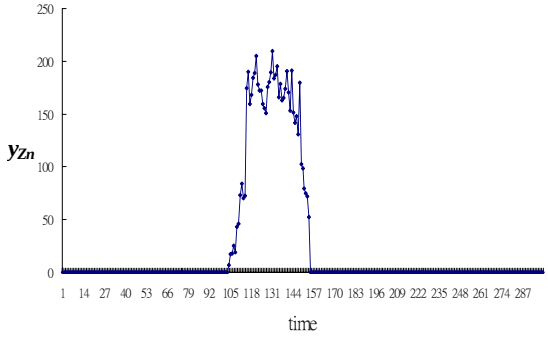

Fig. 16 y_{Zn} detected at network management unit Z under DDoS

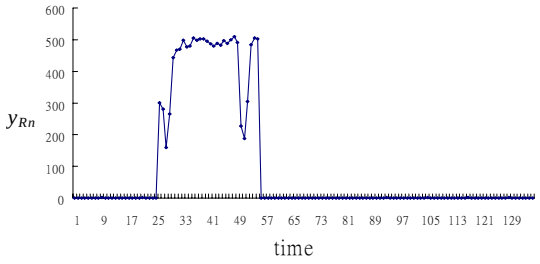

Fig. 17 y_{Rn} detected at network management unit R during DRDoS

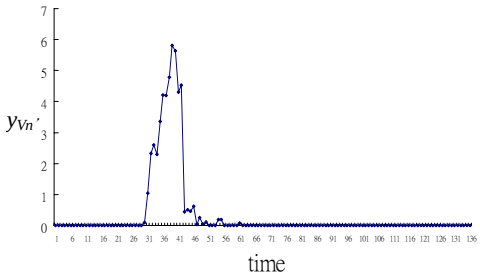

Fig. 18 y_{Vn}' detected at network management unit V under DRDoS

(3) Experiment 4: Detection Timings

Table 6 lists the details of attacks issued in experiment 4. The nodes used to detect an

attack are equipped by Pentium IV 1.8 G CPU, 1,024 MB memory, and 10M/100M network interface.

Table 6 Attack details in which a packet (a TCP SYN or other TCP packets) was 255 bytes in length.

Attack	Type of attack packets issued	Attack duration	Attack intensity
TCP SYN flood	TCP SYN packets with randomly generated source IPs	10 sec	13,000 pkts/sec (=3.315 MB/sec)
Reflective TCP SYN flood	TCP SYN packets with limited number of source IPs	10 sec	13,000 pkts/sec (=3.315 MB/sec)
Reflective TCP flood	TCP packets, other than TCP SYN (e.g., RST, ACK….), with limited number of source IPs	10 sec	13,000 pkts/sec (=3.315 MB/sec)

This experiment evaluated the following security systems (IDSs and firewalls): Snort v.1.8.7, FG-60 (by FortiNet); Kaspersky Anti-Hacker 1.5 (Kaspersky in short, by Kaspersky Labs); McAfee VirusScan Home Edition 7.0 (McAfee VirusScan in short, by McAfee, Inc.); Panda Titanium Antivirus 2005 (Panda Titanium in short, by Panda Software); SNIDS (Single-node IDS) [1], which locates bandwidth consumption attacks [31] for a node by computing its data density, TS / x, and data occupation rate, $TS / bandwidth$, where TS is the total traffic volume transmitted to the node within the period of time x (in msec) and $bandwidth$ is the bandwidth provided by the subnet, e.g., $subnet_j$, on which the node is located. Resource consumption attacks were detected by calculating the node's request frequency, NR / x, and request occupation rate, NR / NP, where NR is the number of request packets transmitted to the node within the period of time x (in msec) and NP is the number of packets transmitted to $subnet_j$.

Table 7 Detection results of TCP SYN floods at attack intensity=13,000 pkts/sec, a packet=255 bytes, parameters of IDeFT are a=500 packets, h=1000 packets, N=2000 and N=5000 packets.

Statistics / Secu. Systems	ART/SD (sec)	ABRT/SD (sec)	ATT/SD (sec)	AWT/SD (sec)	Nodes deployed
Snort	2.2/0.04	--	10.0/0.03	--	Victim
FG-60	1.2/0.04	--	10.0/0.0	--	FG-60 hardware
Kaspersky	10.0/0	--	10.0/0.0	--	Victim
McAfee VirusScan	2.0/0	--	10.0/0.0	--	Victim
Panda Titanium	2.0/0	--	10.0/0.0	--	Victim
SNIDS	6.3/0.02	--	13.2/0.02	2.89/0.02	Victim
IDeFT (on a=500, h=1000, and N=2000 pkts)	0.3/0.003	0.0/0.0	10.0/0.0	0.0/0.0	Zombie
	0.51/0.005	0.0/0.0	10.0/0.0	0.0/0.0	Victim
IDeFT (on a=500, h=1000, and N=5000 pkts)	0.53/0.004	0.0/0.0	10.0/0.0	0.0/0.0	Zombie
	0.75/0.005	0.0/0.0	10.0/0.0	0.0/0.0	Victim

We removed most of Snort's option tree nodes [50] since they are used to detect content-based attacks, e.g., buffer overflow and IIS intrusions. Only rule tree nodes and those option tree nodes that detect "logical" DoS attacks [51] were used.

Table 7 lists the results of the detection on TCP SYN floods at attack intensity=13,000 pkts/sec. The items considered included: (1) average response time (ART) where response time is the time period from the time point an attack begins to the time the attack is discovered; (2) average blocking response time (ABRT) where blocking response time is the time period from the moment an attack is discovered to the time point an attack packet is first blocked; (3) average turnaround time (ATT) where turnaround time is the period from the beginning of an attack to the time point detection ends; and (4) average waiting time (AWT)

where waiting time is the period from when a packet arrives at a node to the time point the packet begins to be analyzed. SD represents the standard deviation obtained from twenty-fold experiments. ART generated by the IDeFT was the shortest, and it detected an attack almost immediately, i.e., AWT=0. However, except for the IDeFT (SNIDS and IDeFT), all other systems addressed do not support measuring mechanisms to evaluate ABRT (AWT).

Table 8 Detection results of reflective TCP SYN floods at attack intensity =13,000 pkts/sec.

Statistics / Secu. Systems	ART/SD (sec)	ABRT/SD (sec)	ATT/SD (sec)	AWT/SD (sec)	Nodes deployed
Snort	1.09/0.0	--	10.0/0.02	--	Victim
FG-60	1.01/0.0	--	10.0/0.0	--	FG-60 hardware
Kaspersky	10.0/0.0	--	10.0/0.0	--	Victim
McAfee VirusScan	2.0/0.0	--	10.0/0.0	--	Victim
Panda Titanium	2.0/0.0	--	10.0/0.0	--	Victim
SNIDS	5.8/0.01	--	12.9/0.03	2.8/0.02	Victim
IDeFT on a=500, h=1000, and N=2000 packets	0.32/0.005	0.0/0.0	10.0/0.0	0.0/0.0	Zombie
	0.79/0.007	0.0/0.0	10.0/0.0	0.0/0.0	Reflector
	0.58/0.004	0.0/0.0	10.0/0.0	0.0/0.0	Victim
IDeFT on a=500, h=1000, and N=5000 packets	0.58/0.006	0.0/0.0	10.0/0.0	0.0/0.0	Zombie
	0.97/0.009	0.0/0.0	10.0/0.0	0.0/0.0	Reflector
	0.81/0.006	0.0/0.0	10.0/0.0	0.0/0.0	Victim

The data concerning FG-60 was obtained through MS Internet Explorer rather than via its console. The SNIDS was a M/M/1/k model [52,53]. Its μ 12,400 pkts/sec, $\lambda = 13{,}000$ pkts/sec, i.e., $\rho = 1.048387$, and buffer size=32 packets. Theoretically, its average queue length $L = 20.12$ packets, $L_q = 19.13$ packets, $W = 1.64$ ms and $W_q = 1.56$ ms. However, our AWT definition is the period of time from when a packet arrives at a detector to the time point it begins to be analyzed.

$$AWT = \frac{(NDP * RDT + \lambda * W_q)}{\lambda} \tag{11}$$

where *NDP* is the number of dropped packets per second, and *RDT* is the retransmission delay of a dropped packet which is set to 60 sec. Theoretically, AWT=2.543 sec. Its measured value, as shown in Table 7, was 2.89 sec. If buffer size=64 packets, then AWT would be 2.398 sec. Its measured value would then be 2.61 sec. IDeFT only counts number of arriving packets, its μ λ. Therefore, $NPD = 0$, $W_q = 0$ and then $AWT = 0$.

Table 8 lists the results of detecting reflective TCP SYN floods. IDeFT only spent 0.32, 0.79 and 0.58 seconds to detect an attack at zombie, reflector and victim, respectively, on a=500, h=1000 and N=2000 packets. When N=5000, it took longer detection time. The detection results of reflective TCP floods are listed in Table 9. The waiting times shown in both tables were also calculated by (11), even though some types of packets sent do not retransmit a dropped packet.

Table 9 Detection results of reflective TCP floods at attack intensity=13,000 pkts/sec.

Statistics / Secu. Systems	ART/SD (sec)	ABRT/SD (sec)	ATT/SD (sec)	AWT/SD (sec)	Nodes deployed
Snort	1.02/0.0	--	10.0/0.01	--	Detector1
FG-60	1.05/0.0	--	10.0/0.0	--	FG-60 hardware
Kaspersky	10.0/0.0	--	10.0/0.0	--	Detector1
McAfee VirusScan	2.0/0.0	--	10.0/0.0	--	Detector1

Panda Titanium	2.0/0.0	--	10.0/0.0	--	Detector1
SNIDS	6.6/0.01	--	12.8/0.01	2.6/0.01	Detector1
IDeFT on a=500, h=1000, and N=2000 packets	0.38/0.002	0.0/0.0	10.0/0.0	0.0/0.0	Zombie
	0.51/0.004	0.0/0.0	10.0/0.0	0.0/0.0	Reflector
	1.01/0.007	0.00.0	10.0/0.0	0.0/0.0	Victim
IDeFT on a=500, h=1000, and N=5000 packets	0.53/0.003	0.0/0.0	10.0/0.0	0.0/0.0	Zombie
	0.72/0.005	0.0/0.0	10.0/0.0	0.0/0.0	Reflector
	1.08/0.008	0.0/0.0	10.0/0.0	0.0/0.0	Victim

(4) Experiment 5: Detection Accuracy

The purpose of experiment 5 was to evaluate detecting accuracy. A TCP SYN flood attacking tool was deployed to flood the victim 500 times, with each attack lasting 1-5 minutes issuing SYN or other TCP packets ranging from 500 to 25000 per second due to the reasons stated in [54]. By mixing with flooding attacks, we sent normal traffic from zombie to the reflectors and victim 500 times. Each also lasted one to five minutes. Table 10 shows that both victim's and zombie's detecting accuracies were 100%. The results with SNIDS was 99.65%. The error was from low attack intensities being treated as normal traffic.

Table 10 Detection accuracies of detecting TCP SYN flood attacks

Statistics / Secu. Systems	Detection location	True Positives	True Negatives	Detection Accuracy
Snort	Victim	89.5%	95.2%	92.35%
FG-60		95.4%	94.5%	94.95%
Kaspersky		99.2%	100%	99.60%
McAfee VirusScan		89.5%	99.3%	94.40%
Panda Titanium		87.7%	98.9%	93.30%
SNIDS		99.3%	100%	99.65%
IDeFT		100%	100%	100%
	Zombie (attacker)	100%	100%	100%

In a reflective TCP SYN flood attack, reflectors from three different subnets are employed. The detecting accuracies of both victim and zombie as shown in Table 11 were also 100%. But the true negative of the reflector in subnet 1 is 97.9% in average since the detection of reflectors absolutely relies on the incoming RST packets sent from a victim (recall, formula (8)). However, when flooded, a victim may temporarily lose its detecting capability, and thus be unable to reply with RST packets to reflectors, consequently causing some level of false negatives.

Finally, in a reflective TCP flood, attackers send packets to reflectors' closed ports, resulting in replying with RST packets to flood the victim. The detecting accuracies of zombie, reflectors and victim as shown in Table 12 are all 100%.

Table 11 Detection accuracies of detecting reflective TCP SYN flood attacks

Statistics / Secu. Systems	Detection location	True Positives	True Negatives	Detection Accuracy
Snort	Victim	95.6%	94.4%	95%
FG-60		95.8%	94.7%	95.25%
Kaspersky		99.5%	97%	98.25%
McAfee VirusScan		90.3%	96.5%	94.40%
Panda Titanium		88.1%	95.8%	91.95%
SNIDS		100%	97%	98.5 %
IDeFT		100%	100%	100%
	Reflector in subnet 1	100%	97.9%	98.95%
	Reflector in subnet 2	100%	100%	100%
	Reflector in subnet 3	100%	100%	100%
	Zombie (attacker)	100%	100%	100%

Table 12 Detection accuracies of detecting reflective TCP flood attacks

Statistics / Secu. Systems	Detection location	True Positives	True Negatives	Detection Accuracy
Snort	Victim	91.3%	96.3%	93.80%
FG-60		95.8%	94.8%	95.30%
Kaspersky		98.8%	98.1%	98.45%
McAfee VirusScan		90.1%	98.3%	94.20%
Panda Titanium		89.9%	97.4%	93.65%
SNIDS		98.8%	100%	99.40%
IDeFT		100%	100%	100%
	Reflector in subnet 1	100%	100%	100%
	Reflector in subnet 2	100%	100%	100%
	Reflector in subnet 3	100%	100%	100%
	Zombie (attacker)	100%	100%	100%

VI. Conclusions and Future Research

In recent research, many schemes have been proposed to solve DoS/DDoS problems, but only a few are now in use. Currently, a host under DoS/DDoS attack can contact its Internet service provider (ISP) or upstream ISP to halt services for a period of time, e.g., half an hour, to protect the host from attacks. But this may seriously affect normal services. Also, DDoS threats are getting stronger day by day since attackers compromise distributed resources to flood victims. The final solution is to shut down the victim and local reflectors, and/or to coordinate with collaborative network management units to power off or shut down their local reflectors. However, united defense seems able to defeat such attacks effectively and efficiently. Attackers, reflectors and victims are detected at a node at the same time instead of at a router. Therefore, the damage caused by an attack is limited.

Sentinel provides a forecast mechanism to detect intrusion. Alerting message is then sent either in-band or out-band so that copro_unit can respond appropriately in advance. Through the cooperation among response manager, tracer and sentinel, copro_unit can filter the malicious packets by simply checking their signal marking bits without affecting other legal services. We also employ certification authority to authenticate communication for response manager when tracer attempts to trace intrusion paths so that hosts can be protected from being compromised by experienced intruders.

However, holding and processing forwarded packets by sentinel delays network transmission. Therefore, a detector should act as an NIDS rather than an HIDS. Certification authority and register authority are deployed to make IDeFT satisfy the third and fourth guiding principles of a forecast model.

In our scheme, each node in a network management unit is installed a CIDS. With customized CUSUM parameters, traditional and reflective TCP SYN floods and other TCP flood attacks in a network management unit can be completely monitored with the decision function $d_n(y_n)$ to real-time detect the role a host may play. These attacks can be defeated in their initial stage because a zombie can be identified by its local CIDS. The detection of reflectors is helpful in overcoming traceback system's tracing limits. Intrusion paths from the zombie to reflectors can then be recognized. However, an attacker of a spoofed or non-spoofed logical attack can be traced directly.

With the assistance of a lightweight detecting approach, like CUSUM, and the help of traceback and filtering, we can mitigate the impact resulted from an attack. CIDS offers us more defending power against the distributed attack to ensure a more secure Internet. Furthermore, only response manager has public IP but is protected by a security mechanism. Other components of the scheme are transparent to users so that nobody can compromise them.

Our future research includes implementing sentinel with the Linux bridge and putting it in front of the physical interface (Internet side) of an edge router. Therefore, egress packets will directly flow through sentinel without any redirection. This occurrence will improve system performance. But a powerful hardware platform is required to reduce transmission delay. Besides, an authenticated IP marking mechanism may have to be included to protect in-band alerting message. IDeFT's reliability, cost and performance models will also be derived and evaluated in our future research.

VII. References

[1] Leu, F.Y. and Lin, J.C., "A Performance-Based Grid Intrusion Detection System," Proceedings of the International Computer Software and Applications Conference, 2005, pp.525-530.

[2] Darmohray, T. and Oliver, R., "Hot Spares for DoS attacks,"USENIX ;login:, vol. 25, no. 7, July 2000. http://www.usenix.org/publications/login/2000-7/apropos.html

[3] Thottan, M. and Ji, C.Y., "Anomaly Detection in IP networks," IEEE Transactions on Signal Processing, vol.51, no.8, Aug. 2003, pp.2191-2204.
[4] Mukkamala, S. and Sung, A.H., "Artificial Intelligent Techniques for Intrusion Detection," Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, vol.2, Oct. 2003, pp.1266-1271.
[5] Netscreen Technologies Inc., "Intrusion Detection and Prevention," A White Paper, http://www.netscreen.com
[6] Xue, Q., Sun, J.Z. and Wei, Z., "TJIDS: An Intrusion Detection Architecture for Distributed Network," Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering, 2003, pp.709-712.
[7] Rivest, R.L. and Lampson, B., SDSI: A Simple Distributed Security Infrastructure, http://theory.lcs.mit.edu/~rivest/sdsi.htm
[8] Bellovin, S.M., "Distributed Firewalls," USENIX ;login:, Nov. 1999, pp.37-39. http://www.cs.columbia.edu/~smb/papers/distfw.html
[9] Leu, F.Y., Yang, W.J. and Chang, W.K., "IFTS: Intrusion Forecast and Traceback based on Union Defense Environment," Proceedings of the International Conference on Parallel and Distributed Systems, 2005, pp. 716- 722.
[10] Humphrey, M., Thompson, M.R. and Jackson, K.R., "Security for Grids," Proceedings of the IEEE, vlo.93, no.3, 2005, pp. 644-652.
[11] Wang, H., Zhang, D. and Shin, K.G., "Change-Point Monitoring for the Detection of DoS Attacks," IEEE Transactions on Dependable and Secure Computing, vol.1, no.4, Oct.-Dec. 2004, pp.193-208.
[12] Gibson, S., "DRDoS, Distributed Reflection Denial of Service" http://grc.com/dos/drdos.htm
[13] Moore, D., Shannon, C., Brown, D., Voelker, G.M. and Savage, S., "Inferring Internet Denial of Service Activity," the USENIX Security Symposium, Aug. 2001. ACM Transactions on Computer Systems (TOCS), May 2006. http://www.caida.org/publications/papers/2006/backscatter_dos/backscatter_dos.pdf
[14] Brodsky, B.E. and Darkhovsky, B.S., *Nonparametric Methods in Change-point Problems*, Kluwer Academic Publishers, 1993.
[15] Wang, H., Zhang, D. and Shin, K.G., "Detecting SYN Flooding Attacks," Proceedings of the Joint Conference of the IEEE Computer and Communications Societies, 2002, pp.1530-1539.
[16] Paxson, V., "An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks," Computer Communication Review, 31(3), July 2001, pp.38-47.
[17] S. S.H., Huang, Lychev, R. and Yang, J.H., "Stepping-Stone Detection via Request Response Traffic Analysis," Proceedings of the International Conference on Autonomic and Trusted Computing, Lecture Notes in Computer Science, vol. 4610, 2007, pp. 276-285.
[18] Chang, R.K.C., "Defending against Flooding-based Distributed Denial-of-Service Attacks: a Tutorial," IEEE Communications Magazine, vol.40, Oct. 2002, pp.42-51.
[19] Leu, F.Y., Cheng, J.J. and Hung, C.H., "UDAIDTS：Union Defense with Active Intrusion Detection and Hash-based Traceback System," Proceedings of the International Conference on Information Management, Taiwan, 2003, pp.1090-1097. (in Chinese)
[20] Gibson, S., "The Strange Tale of the Denial of Service Attacks Against GRC.COM," http://grc.com/dos/grcdos.htm, March 2002.
[21] Ferguson, P. and Senie, D., "Network Ingress Filtering: Defeating Denila of Service Attacks Which Employ IP Source Address Spoofing," RFC 2827, May 2000.
[22] Carl, G. Kesidis, G., Brooks, R.R. and Rai, S., "Denial-of-Service Attack-Detection Techniques," the IEEE Internet Computing, Jan./Feb. 2006, pp. 82-89.

[23] Bellovin, S.M., "ICMP Traceback Messages," IETF draft, March 2000; http://www.research.att.com/smb/papers/draftbellovin -itrace-00.txt
[24] Savage, S. et al., "Network Support for IP Traceback," the IEEE/ACM Transaction on Networking, vol.9, no.3, June 2001, pp.226-237.
[25] Snoeren, A.C. et al., "Single-Packet IP Traceback," the IEEE/ACM Transaction on Networking, vol.10, no.6, Dec. 2002, pp.721-734.
[26] Belenky, A. and Ansari, N., "On IP Traceback," the IEEE Communications Magazine, vol.41, July 2003, pp.142-153.
[27] Dawn Song, X.D. and Perrig, A., "Advanced and Authenticated Marking Schemes for IP Traceback," Proceedings of the Joint Conference of the IEEE Computer and Communications Societies, vol.2, 2001, pp.878-886.
[28] Jin, G. and Yang, J.G., "Deterministic Packet Marking based on Redundant Decomposition for IP traceback," the IEEE Communications Letter, vol.10, no.3, March 2006. http://www.comsoc.org/livepubs/comml/public/2006/mar/comml_jin.html
[29] Gao, Z.Q. and Ansari, N., "Tracing Cyber Attacks from the Practical Perspective," the IEEE Communications Magazine, May 2005, pp.123-131.
[30] Stone, R., "Centertrack: An IP Overlay Network for Tracking DoS Floods," the USENIX Security Symposium, 2000, pp.199-212.
[31] Chang, H.Y. et al., "Deciduous: Decentralized Source Identification for Network-Based Intrusions," Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management, 1999, pp.701-714.
[32] Burch, H. and Cheswick, B., "Tracing Anonymous Packets to Their Approximate Source," the USENIX LISA, 2000, pp.319-327.
[33] Mirkovic, J., Prier, G. and Reiher, P., "Attacking DDoS at the Source," Proceedings of the IEEE International Conference on Network Protocols, 2002, pp.312-321.
[34] Gil, T.M. and Poletter, M., "MULTOPS: A Data-Structure for Bandwidth Attack Detection," Proceedings of the USENIX Security Symposium, 2001.
[35] Forrest, S., Longstaff, T.A, Hofmeyr, S.A. and Somayaji, A., "A Sense of Self for Unix Processes," Proceedings of the IEEE Symposium on Security and Privacy, 1996, pp. 120-128.
[36] Blanc, M., Oudot, L., and Glaume, V., Global Intrusion Detection: Prelude Hybrid IDS, http://www.exaprobe.com/labs/downloads/Papers/Prelude.pdf
[37] Luo, J. Pan, Z.S., Ni, G.Q. and Hu, G.U., "Research on Cost-Sensitive Learning in One-Class Anomaly Detection Algorithms," Proceedings of the International Conference on Autonomic and Trusted Computing, Lecture Notes in Computer Science, vol. 4610, 2007, pp. 259-268.
[38] Abdullah, K., Lee, C., Conti, G., Copeland, J.A., "Visualizing Network Data for Intrusion Detection," Proceedings of the IEEE Workshop on Information Assurance Workshop, 2005, pp. 100-108.
[39] Yu, J.Q., Reddy, Y.V.R., Selliah, S., Kankanahalli, S., Reddy S. and Bharadwaj, V., "TRINETR: An Intrusion Detection Alert Management System," SIP Lab, Concurrent Engineering Research Center Lane Department of Computer Science and Electrical Engineering, June 2004, pp. 235-240.
[40] Yin, Q., Shen, L., Zhang, R. and Li, X., "A New Intrusion Detection Method Based on a Behavioral Model," World Congress on Intelligent Control and Automation, June 2004, pp. 4370-4374.
[41] Chau, M., Xu, J.J. and Chen, H., "Extracting Meaningful Entities from Police Narrative Reports," Proceedings of National Conference on Digital Government Research, 2002, pp. 271-275.

[42] Cabrera, J.B.D., Lewis, L., and Mehra, R.K., “Detection and Classification of Intrusion and Faults Using Sentences of System Calls,” SIGMOD Record, vol.30, no.4, 2001, pp.25-34.
[43] Leu, F.Y. and Hu, K.W., “A Real-Time Intrusion Detection System using Data Mining Technique,” Journal of Systemics, Cybernetics and Informatics, vol. 6, no. 2, 2008 (online). www.iiisci.org/journal/CV$/sci/pdfs/T171GTB.pdf
[44] Benlenky, A. and Ansari, N., “Tracing Multiple Attackers with Deterministic Packet Marking (DPM),” Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, 2003, pp. 49-52.
[45] Benlenky, A. and Ansari, N., “Accommodating Fragmentation in Deterministic Packet Marking for IP Traceback,” Proceedings of the IEEE Global Communications Conference, 2003, pp. 1374-1378.
[46] Leu, F.Y. and Lin, J.C., “Integrating Grid with Intrusion Detection,” Proceedings of the IEEE 19th International Conference on Networking and Applications, 2005, pp.304-309.
[47] Peng, T., Leckie, C. and Ramamohanarao, K., “Detecting Reflector Attacks by Sharing Beliefs,” Proceedings of the IEEE Global Telecommunications Conference, 2003, pp.1358-1362.
[48] Paxson, V., “An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks,” the ACM Computer Communications Review, vo.31, no.3, July 2001.
[49] Thompson, K., Miller, G.J., and Wilder, R., “Wide-Area Internet Traffic Patterns and Characteristics,” the IEEE Network, vol.11, no.6, Nov./Dec. 1997. http://www.comsoc.org/ni/private/1997/nov/Thompson.html
[50] http://www.samspublishing.com/articles/article.asp?p=31745&seqNum=3&rl=1
[51] http://support.microsoft.com/kb/q267559/
[52] Gross, D. and Harris, C.M., Fundamentals of Queueing Theory, 3rd ed., John Wiley & Sons, 1998.
[53] Kleinrock, L., Queueing Systems, Volume 1: Theory, John Wiley & Sons, 1975.
[54] Oliver, R., “Countering SYN Flood Denial-of-Service Attacks,” Tech Mavens, Inc., Aug. 2001. http://www.tech-mavens.com/synflood.htm