CHAPTER 1

INTRODUCTION

Denial of Service attacks were first used to “have fun”, get some kind of revenge from system operators or make complex attacks possible, such as blind spoofing on services. IRC servers were also often targeted after one got insulted on a channel. At this time networks and Internet uses were “confidential”, and those attacks had very limited impact. With time and as the Internet gets more and more used as a communication channel, hacktivism becomes more and more popular. Geopolitical situations, wars, religious concerns, ecology, any motive is then good to launch attacks on companies, political organization or even national IT infrastructures. A more recent use of Denial of Service is linked to online gaming.

Many servers have been victims of such attacks, generated by unhappy gamers who lost lives or their favorite weapon during game. However, the very use of Denial of Service today is definitely extortion. More and more enterprises rely on their IT infrastructure. Mail, critical data and even phone are handled by the network. Very few companies can survive without their main communication channel. Old single source attacks are now countered easily by many defense mechanisms and the source of these attacks can be easily rebuffed or shut down with improved tracking capabilities. However, with the astounding growth of the Internet during the last decade, an increasingly large number of vulnerable systems are now available to attackers. Attacker can now employ a large number of these vulnerable hosts to launch an attack instead of using a single server, an approach that is not very effective and detected easily. A distributed denial of service (DDoS) attack is a large-scale, coordinated attack on the availability of services of a victim system or network resources, launched indirectly through many compromised computers on the Internet.

To launch a DDoS attack, the attacker first establishes a network of computers that will be used to generate the huge volume of traffic needed to deny services to legitimate users of the victim. To create this attack network, attackers discover vulnerable hosts on the network. Vulnerable hosts are either those that are running no antivirus or out-of-date antivirus software, or those that have not been properly patched. These are exploited by the attackers who use the vulnerability to gain access to these hosts. The next step for the attacker is to install new programs on the compromised hosts of the attack network. The hosts running these attack tools are known as zombies, and they can be used to carry out any attack under the control of the attacker. Numerous zombies together form an army.

There are two categories of DDoS attacks, typical DDoS attacks and DRDoS (Distributed Reflection Denial of Service) attacks. In a typical DDoS attack, the master computer orders the zombies to run the attack tools to send huge volume of packets to the victim, to exhaust the victim’s resources. Unlike the typical DDoS attacks, the army of a DRDoS attack consists of master zombies, slave zombies, and reflectors.

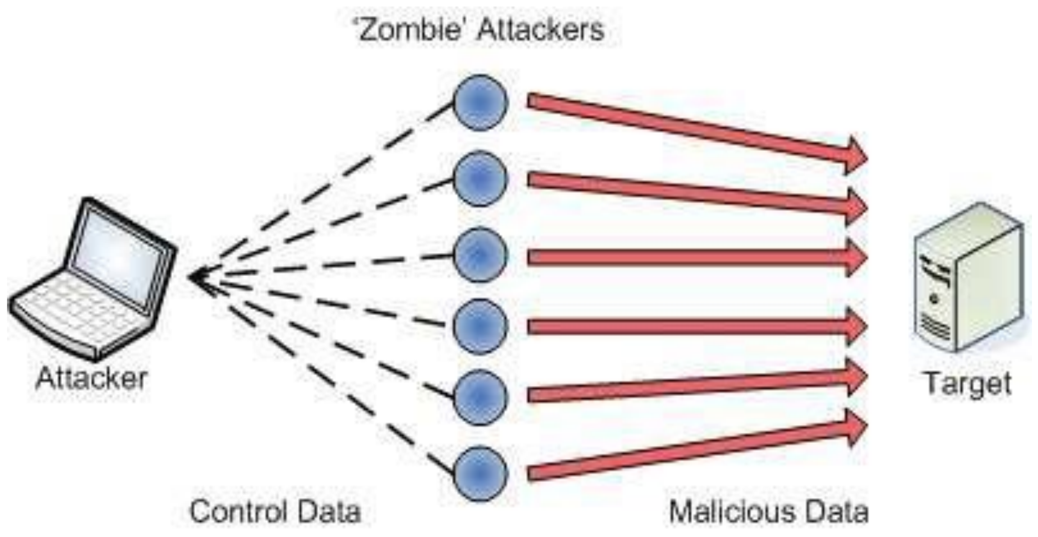


Fig 1.1 Typical DDoS Attack

Distributed Denial-of-Service (DDoS) attacks are a critical threat to the Internet. DDoS attacks are targeted at exhausting the victim's resources, such as network bandwidth, computing power, and operating system data structures. To launch a DDoS attack, the attacker first establishes a network of computers that will be used to generate the huge volume of traffic needed to deny services to legitimate users of the victim. To create this attack network, attackers discover vulnerable hosts on the network. Vulnerable hosts are either those that are running no antivirus or out-of-date antivirus software, or those that have not been properly patched. These are exploited by the attackers who use the vulnerability to gain access to these hosts. The next step for the attacker is to install new programs on the compromised hosts of the attack network. The hosts running these attack tools are known as zombies, and they can be used to carry out any attack under the control of the attacker. However, the memory-less feature of the Internet routing

mechanisms makes it extremely hard to trace back to the source of these attacks. Powerful DDoS toolkits are available to potential attackers, and essential networks are ill prepared for defense. The security community has long known that DDoS attacks are possible, but only in the past three years have such attacks become popular with hackers.
The difference in this type of attack is that slave zombies are led by master zombies to send a stream of packets with the victim's IP address as the source IP address to other uninfected machines (known as reflectors), exhorting these machines to connect with the victim. Then the reflectors send the victim a great volume of traffic, as a reply to its exhortation for the opening of a new connection, because they believe that the victim was the host that asked for it.

1.1 DDoS attacks techniques

From time to time there are various techniques used to cause the attack. Some of the important type of attack techniques is as follows:

1.1.1 Network protocols attacks

These attacks aim at the transmission channel, and therefore target the IP stack, which is an entry point for critical resources such as memory and CPU.

1.1.2 SYNFloods

SYNFloods are typical concept-based denial of service attacks as they entirely rely on the way TCP connections are established. During the initial 3-way handshake, the server fills up a table, the TCB (Transmission Control Block), which keeps session information in memory. When a server receives the initial SYN packect from a client, it sends backs a SYN-ACK packet and creates an entry in the TCB. The connection is in a TIME_WAIT status, for as long as the server waits for the final ACK packet from the client. If this final ACK packet is not received, another SYN-ACK is sent to the client. Finally, if after multiple retries none of the SYN-ACK packets are acknowledged by the client, the session is closed and flushed from the TCB. The period between the transmission of the first SYN-ACK and the closure of the session is usually around 30 seconds. During this period it is possible to send hundreds of thousands of SYN packets to an open port and never acknowledge the SYN-ACK packet of the server. The TCB is quickly overloaded and the stack does not accept any new connections and existing ones are dropped.

1.1.3 UDP Flood

UDP is also naturally bound to be a vector of Denial of Service attacks. As it is specified, a server receiving a UDP packet on a closed port sends back an ICMP Port Unreachable packet to the source. The data part of the ICMP packet is filled with at least the first 64 bytes of the original UDP packet. As no limit or quota is specified as a standard, it is then possible to send huge amount of packets on closed ports. At very high load, operations necessary to generate ICMP error packets consume a lot of CPU, eventually leading to CPU resource exhaustion.

1.1.4 Layer 2 connectivity attacks

The first and most obvious, solution to generate a denial of service on layer 2 connectivity is to target internetworking devices, usually switches. ARP Flooding attacks, once used to have switches behave like hubs, are still efficient as some switches do crash when their switching table is filled up. Then sending hundreds of thousands of frames with random source MAC address will usually be enough to prevent any communication from and/or to hosts connected to the switch. If a target valid MAC address is provided it is then possible that all the switches between the attacker and the system with the target MAC address will be downed. On wired networks ARP mechanisms are also an excellent vector. This resolution protocol makes black holing attacks, which consist in redirecting the traffic to nowhere or to a silent node, possible in different manner.

1.1.5 Nuke

A Nuke is an old denial-of-service attack against computer networks consisting of fragmented or otherwise invalid ICMP packets sent to the target, achieved by using a modified ping utility to repeatedly send this corrupt data, thus slowing down the affected computer until it comes to a complete stop. A specific example of a nuke attack that gained some prominence is the WinNuke, which exploited the vulnerability in the NetBIOS handler in Windows_95. A string of out-of-band data was sent to TCP port 139 of the victim's machine, causing it to lock up and display a Blue_Screen_of_Death (BSOD).

1.1.6 Session based attacks

Application connectivity is mostly handled by sessions, usually identified by TCP mechanisms. The number of simultaneous sessions is an important factor that affects the performance of a given application and, as such, has to be limited. When this limitation is simply based on network and transport information (IP+TCP), it appears that generating a denial of service is trivial. A simple attack opening TCP sessions and leaving them open will quickly fill-up all available session slots, preventing any new session from being established. This pending session attack is a layer 7 equivalent of the SYNFlood. However, when gigabits of traffic may be necessary at layer 4, only a few thousands of packets sent within a few seconds are necessary to block any new session establishment.

Full and legitimate sessions can also damage applications, starting with the F5 attack, which consisted in keeping the F5 key pressed to force a complete refresh of the web page loaded on Internet Explorer. With this old and trivial attack, server resources could get exhausted simply because of the number of web pages to serve. This kind of session flooding attack can also damage other critical paths of the communication channel:

− *Telecom link*: the volume of data transferred from the server to the client can fill-up the link to the Internet. In this case no communication is possible through this link; the focused denial of service attack becomes global;

− *Server application*: high number of connections can reach the limit of authorized simultaneous session, the attack becomes similar to a pending session attack. If the limit is not set handling of a high volume of sessions may consume most of the system resources.

− *Third-party application*: Most applications are linked to middleware and databases. In each case, possible bottlenecks can appear as those third party applications may encounter internal problems handling the huge amount of requests sent by the original application. These effects can also be the consequence of internals weaknesses.

As ominous as the threat is today, it will only worsen as tools are built to evade defenses. Soon, DDoS floods will appear that are difficult to distinguish from legitimate traffic, and packet rates from individual flood sources will be low enough to escape notice by local administrators. To meet the increasing need for detection and response, researchers face these major issues:

- A stand-alone router on the attack path should automatically recognize that the network is under attack and adjust its traffic flow to ease the attack impact downstream.
- The detection and response techniques should be adaptable to a wide range of network environments, preferably without significant manual tuning.

- Attack detection should be as accurate as possible. False positives can lead to inappropriate responses that cause denial of service to legitimate users. False negatives result in attacks going unnoticed.
- Attack response should employ intelligent packet discard mechanisms to reduce the downstream impact of the flood while preserving and routing the non-attack packets.
- The detection method should be effective against a variety of attack tools available today and robust against future attempts by attackers to evade detection.

These are demanding goals, but there are several reasons to believe that satisfactory detection and response methods can be designed. DDoS traffic generated by today's tools often has packet crafting characteristics that make it possible to distinguish from normal traffic. For example, in some configurations the Stacheldraht attack tool crafts packets so that the source port is random and the destination port is sequentially increased from one packet to the next Future DDoS tools may include improvements to packet crafting. However, we claim that these tools are unlikely to model legitimate traffic closely enough to produce crafted packets that do not distort statistical measurements of the composition of the traffic. This project proves that relatively simple statistical measures can be used to discriminate DDoS traffic from legitimate traffic in core routers with sufficient accuracy to trace back the attack source IP address and mitigate the effect of the attack downstream.

CHAPTER 2

LITERATURE SURVEY

In fact IP trace back schemes are considered successful if they can identify the zombies from which the DDoS attack packets entered the Internet. Research on DDoS detection [4],[5],[6],[7],[8],[9], mitigation [10],[11],[12] and filtering [13],[14],[15],[16],[17],[18] have been conducted pervasively. However, the efforts on IP trace back are limited. A number of IP trace back approaches have been suggested to identify attackers [19],[20], and there are two major methods for IP trace back, the probabilistic packet marking (PPM)[21],[22],[23],[24] and the deterministic packet marking (DPM) [25],[26],[27],[28]. Both of these strategies require routers to inject marks into individual packets. Moreover, the PPM strategy can only operate in a local range of the Internet (ISP network) where the defender has the authority to manage.

However, this kind of ISP networks is generally quite small, and we cannot trace back to the attack sources located out of the ISP network. The DPM strategy requires all the Internet routers to be updated for packet marking. However, with only 25 spare bits available in as IP packet, the scalability of DPM is a huge problem [22]. Moreover, the DPM mechanism poses an extraordinary challenge on storage for packet logging for routers [29]. Therefore, it is infeasible in practice at present. Further, both PPM and DPM are vulnerable to hacking [30], which is referred to as packet pollution. IP trace back methods should be independent from packet pollution and various attack patterns. In a path breaking approach [31],[32] on DDoS attack detection, researchers compared the packet number distributions of packet flows, which are out of the control of attackers once the attack is launched, and they found that the similarity of attack flows is much higher than the similarity among legitimate flows, e.g. flash crowds.

Entropy rate, the entropy growth rate as the length of a stochastic sequence increases [33], was employed to find the similarity between two flows on the entropy growth pattern [31], and relative entropy, an abstract distance between two probabilistic mass distributions [33], was taken to measure the instant difference between two flows [32]. Understanding the features of DDoS attack is critical for effective attack trace back. However, we have limited real data sets about DDoS attacks. The current knowledge of DDoS attack can be classified as follows: inference based on partial information [34]; real network emulation [35] or simulations [36]; and real attack and defense between two cooperate research groups [37].

It is obvious that hunting down the attackers (zombies), and further to the hackers, is essential in solving the DDoS attack challenge. The summary of the existing DDoS trace back methods can be found in reference [38] and [39]. In general, the trace back strategies are based on packet marking. Packet marking methods include the probabilistic packet marking (PPM) and the deterministic packet marking (DPM). The PPM mechanism tries to mark packets with the router's IP address information by probability on the local router, and the victim can reconstruct the paths that the attack packets went through.

The PPM method is vulnerable to attackers, as pointed out in [30], as attackers can send spoofed marking information to the victim to mislead the victim. The accuracy of PPM is another problem, because the marked messages by the routers who are closer to the leaves (which means far away from the victim) could be overwritten by the downstream routers on the attack tree [21]. At the same time, most of the PPM algorithms suffer from the storage space problem to store large amount of marked packets for reconstructing the attack tree [22], [24]. Moreover, PPM requires all the Internet routers to be involved in marking. Based on the PPM mechanism, Law et al tried to trace back the attackers using traffic rates of packets, which were targeted on the victim [23]. The model bears a very strong assumption: the traffic pattern has to obey the Poisson distribution, which is not always true in the Internet. Moreover, it inherits the disadvantages of the PPM mechanism: large amount of marked packets are expected to reconstruct the attack diagram, centralized processing on the victim, and it is easy be fooled by attackers using packet pollution.

The deterministic packet marking mechanism tries to mark the spare space of a packet with the packet's initial routers information, e.g. IP address. Therefore, the receiver can identify the source location of the packets once it has sufficient information of the marks. The major problem of DPM is that it involves modifications of the current routing software, and it may require every large amount of marks for packet reconstruction. Moreover, similar to PPM, the DPM mechanism cannot avoid pollution from attackers. Reference [24] first introduced the probability-based packet marking method, node appending, which appends each node's address to the end of the packet as it travels from the attack source to the victim. Obviously, it is infeasible when the path is long or there is insufficient unused space in the original packet. The authors proposed the node-sampling algorithm, which records the router address to the packet with probability, *p*, on the routers of the attack path. Then the probability of a packet is marked by a router *d* hops away from the victim is $p(1-p)^{d-1}$. Based on the number of marked packets, we can reconstruct the attack path.

However, it requires large number of packets to improve the accuracy of the attack path reconstruction. Therefore, an edge-sampling algorithm was proposed to mark the start router address and end router address of an attack link and the distance between the two ends. The edge-sampling algorithm fixed the problems of the node-sampling algorithm to some extent. Based on the PPM mechanism, in [23] the traffic which targeted the victim was measured to construct the attack diagram, and then identified where the attackers were located. They focused on the traffic flows which end at the victim and therefore there was a tree, which was rooted at the victim. For a router on the attack tree, the outgoing flow included two parts: the locally generated flows and the transit flows from the upstream router(s) of the attack tree. If *X1* and *X2* are two flows on the attack tree, and *X1*is the upstream flow of *X2*, then $\mathrm{Pr}ob[X > x] \geq \mathrm{Pr}ob[X > x], \forall x$. The victim will collect all the marked packets from the routers and reconstruct the attack tree based on the traffic rates of the different routers. This trace back method heavily depends on the queuing model, and it requires the traffic flows to obey specific patterns, e.g. the Poisson distribution. In [22] the randomize-and-link approach to implement IP trace back based on the probabilistic packet marking mechanism was proposed.

The algorithm targets two aspects: to reconstruct the marks from the marker efficiently and to make the PPM more secure against hackers' pollution. The idea is to have every router X to fragment its unique message *Mx* (e.g. IP address) into several pieces, l M ,M ,...M 0 1 . At the same time, the router calculates the checksum (x) $C = C$ M , named as *cord*. The router assembles the mark as *bi*, and injects *bi* randomly into the unused IPv4 packet header (say *N* bits, which is 25 bits in the paper: 16 bits of fragmentation ID, 1 bit of the fragmentation index, and 8 bits of service type, all of them are used rarely in a common IPv4 packet). The *bi* includes three parts: an index of the pieces (l 2 log bits), a large checksum cord ( ) x $C = C$ M (log | | 2 i $N - l - M$ bits), and a piece of i l i , = 0,1,.., (| | i M bits). The cord is quite large, for example 14 out of 25 bits, therefore, we can treat the cord as a random number, which is hard for hackers to predict. The victim can reconstruct the message efficiently by checking the cord and the index sequence.Yaar, Perrig and Song [40] studied the marking technique to improve the PPM mechanism. They broke the 16 bits marking space into three parts: 1 bit for distance, 2 bits for fragmentation index, and a hash fragmentation of 13 bits. By this modification, the proposed FIT algorithm can trace back the attack paths with high probability after receiving only tens of packets. The FIT algorithm also performed well even in the presence of legacy routers and it is a scalable algorithm for thousands of attack sources. Snoeren et al. proposed a method by logging packets or digests of packets at routers [41], [42]. The packets are digested using bloom filter at all the routers. Based on these logged information, the victim can trace back the leaves on an attack tree. The methods can even trace back a single packet. However, it also places a significant strain on the storage capability of intermediate routers.

In [21] two hybrid schemes which combine the packet marking and packet logging method to trace back the attack sources is proposed - Distributed Link-List Traceback (DLLT) and the Probabilistic Pipelined Packet Marking (PPPM). storage at the intermediate routers. Different from PPM, Dean, Franlin and Stubblefield[26] proposed a deterministic packet marking strategy for IP trace back. Every ingress router writes its own IP address into the outgoing IP packet header, and there is no more marking for the packet. They used an algebraic approach, originally developed for coding theory and learning theory, for encoding trace back information. Their idea is that for any polynomial $f(x)$ of degree d in the prime field $GF(p)$, $f(x)$ can be recovered given $f(x)$ evaluated at $d+1$ unique points. Belenky and Ansari [25] noticed that the PPM mechanism can only solve large flooding attacks, and it is not applicable for attacks comprised of a small number of packets. Moreover, PPM is vulnerable if hackers inject marked packets into the network.

Therefore, the paper proposed a deterministic packet marking method for IP trace back. The basic idea is: at the initial router for an information source, the router embeds its IP address into the packet by chopping the router’s IP into two segments with 17 bits each (16 bits for half of the IP address, and 1-bit works as index). As a result, the victim can trace which router the packets came from. Jin and Yang [27] improved the ID coding of the deterministic packet marking scheme using redundant decomposition of the initial router IP address. For an IP address, they divided them into three redundant segments, 0-13bits, 9-22bits, and 18-31 bits, and then 5 different hash functions are applied on the 3 segments to create 5 results. The resulting 8 segments are recorded in the outgoing packets randomly. The victim can reassemble the source router IP using the packets it has received.

CHAPTER 3

PROBLEM STATEMENT

3.1 Existing trace back methods

A number of IP trace back approaches have been suggested to identify attackers, and there are two major methods for IP trace back, the Probabilistic Packet Marking (PPM) and the Deterministic Packet Marking (DPM).

3.1.1 The Probabilistic Packet Marking method

The PPM mechanism tries to mark packets with the router's IP address information by probability on the local router, and the victim can reconstruct the paths that the attack packets went through.

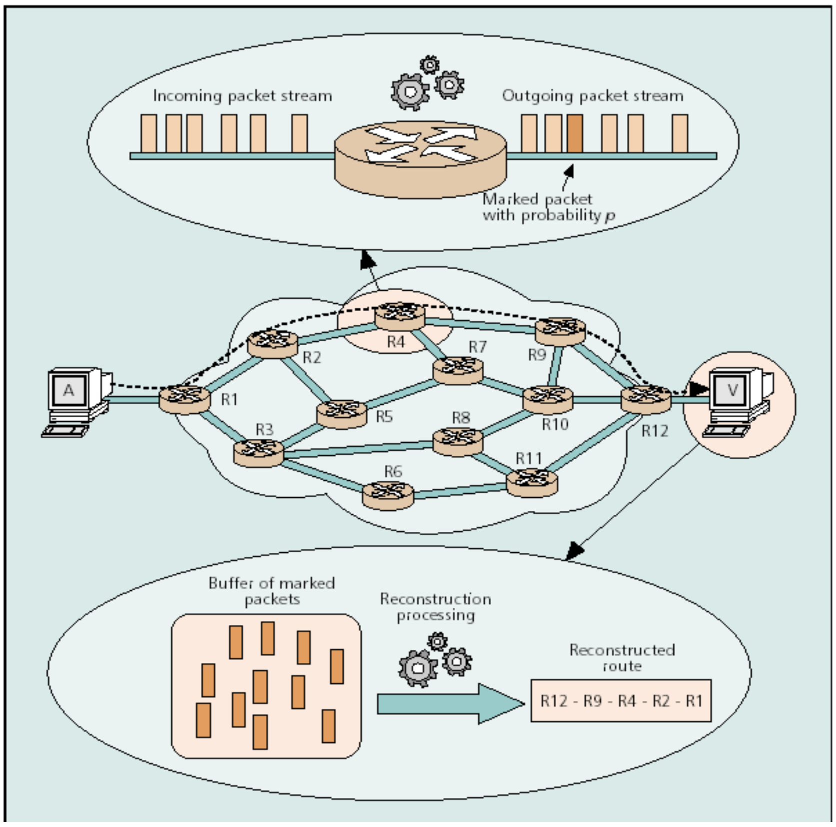


Fig.3.1 PPM

Probabilistic packet marking (PPM) was originally suggested by Burch and Cheswick and was carefully designed and implemented by Savage to solve the IP trace back problem, which can be stated as follows: given a stream of packets arriving at a receiver, identify the source of these packets and the path they took through the network. However, it is apparent that PPM is a general technique (beyond IP trace back) to communicate internal network information to end-hosts. PPM has natural applications in solving the IP trace back problem, which is a potential countermeasure against distributed denial-of-service (DDoS) attacks. In this problem, the internal network information at each router is the IP address of its (incoming) interface and the goal of a PPM scheme for IP trace back is to convey the entire IP-level path from the source to the destination.

A number of potential applications of PPM have since been identified, which include congestion control, robust routing algorithms, dynamic network reconfiguration, and locating Internet bottlenecks. The internal network information in these applications is different from that of IP trace back. For example, in locating bottlenecks and congestion control, the internal network information corresponds to the IP address of the “narrow link” in the network and the level of congestion on this narrow link, respectively. Different PPM schemes differ from each other in the number of PPM bits allocated, the transformation applied at each intermediate router, the number of IP packets required to convey the internal information to the destination, as well as the decoding algorithm performed at the destination. In particular, the two primary measures of the efficiency of a probabilistic packet-marking scheme are the number of PPM bits required in the header of a packet and the number of packets required to convey the internal network information to the destination. When applied to IP networks, reducing the number of PPM bits is a very significant requirement. This is because IP is a mature and globally deployed protocol and has very limited number of available header bits that can be used by PPM applications. In fact, the fewer the number of PPM bits, the greater the number of PPM applications that can be deployed simultaneously in the network.

3.1.2 The Deterministic Packet Marking method

DPM is a packet-marking algorithm. The 16-bit Packet ID field and 1-bit Reserved Flag (RF) in the IP header will be used to mark packets. Each packet is marked when it enters the network. This mark remains unchanged for as long as the packet traverses the network. The packet is marked by the interface closest to the source of the packet on an edge ingress router. The interface makes a distinction between incoming and outgoing packets. Incoming packets are marked; outgoing packets are not marked.

Every incoming packet will be marked. A 32-bit IP address needs to be passed to the victim. A total of 17 bits are available to pass this information: 16-bit ID field and 1-

bit RF. Clearly, a single packet would not be enough to carry the whole IP address in the available 17 bits. Therefore, it will take at least two packets to transport the whole IP address. An IP address will be split into two segments, 16 bits each: segment 0 bits 0 through 15, and segment 1 bits 16 through 31. When a marked packet arrives to the victim, the victim will first determine if the given packet is an attack packet. If it is, the victim would check to see if the table entry for a source address of this packet already exists, and create it if it did not. Then, it would write the segment from the mark, according to the value of the segment number, into the ingress IP address value. After both segments corresponding to the same ingress address have arrived to the destination, the ingress address for a given source address becomes available to the victim.

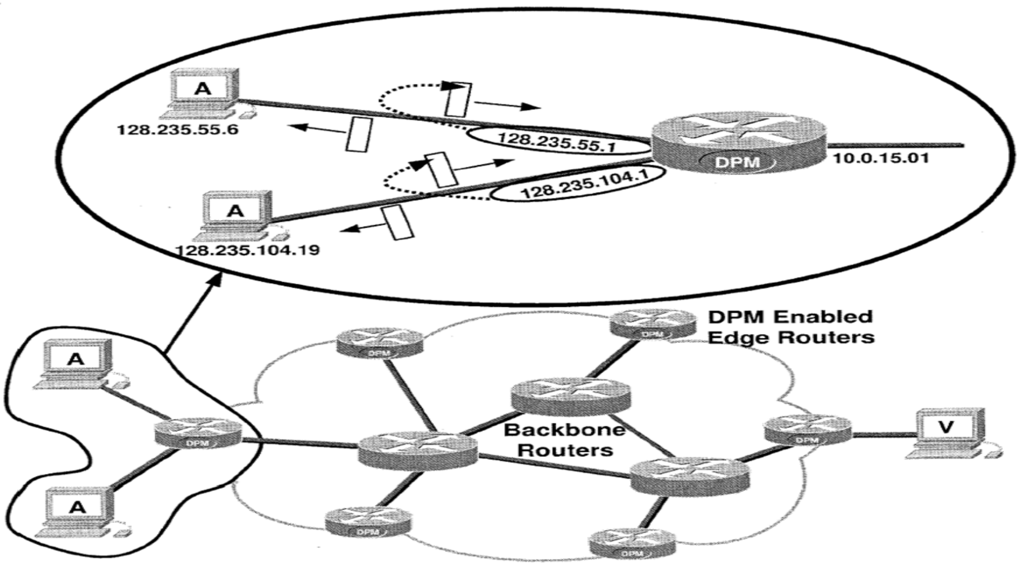


Fig 3.2 DPM

Each packet is marked when it enters the network. Only incoming packets are marked.
Address information of this interface is marked 16 bit ID + 1 bit Flag.

3.2 Limitations of the PPM and DPM methods

However the PPM strategy can only operate in a local range of the Internet where the defender has the authority to manage. However, this kind of ISP networks is generally quite small, and we cannot trace back to the attack sources located out of the ISP network. Further PPM is vulnerable to hacking, which is referred to as packet pollution. IP trace back methods should be independent from packet pollution and various attack patterns. PPM requires update on the existing routing software, which is extremely hard to achieve on the Internet. There are inherited drawbacks of probabilistic packet marking methods such as limited scalability, huge demands on storage space and vulnerability to packet pollutions. Based on the PPM mechanism, Law tried to trace back the attackers using traffic rates of packets, which were targeted on the victim. The model bears a very strong assumption: the traffic pattern has to obey the Poisson distribution, which is not always true in the Internet. Moreover, it inherits the disadvantages of the PPM mechanism: large amount of marked packets are expected to reconstruct the attack diagram, centralized processing on the victim, and it is easy be fooled by attackers using packet pollution. The DPM strategy requires all the Internet routers to be updated for packet marking. However, with only 25 spare bits available in as IP packet, the scalability of DPM is a huge problem. Moreover, the DPM mechanism poses an extraordinary challenge on storage for packet logging for routers. Further DPM is vulnerable to hacking, which is referred to as packet pollution. IP trace back methods should be independent from packet pollution and various attack patterns. DPM requires update on the existing routing software, which is extremely hard to achieve on the Internet. There are inherited drawbacks of probabilistic packet marking methods such as limited scalability, huge demands on storage space and vulnerability to packet pollutions. The deterministic packet marking mechanism tries to mark the spare space of a packet with the packet's initial router's information, e.g. IP address. The major problem of DPM is that it involves modifications of the current routing software, and it may require every large amount of marks for packet reconstruction. Moreover, similar to PPM, the DPM mechanism cannot avoid pollution from attackers.

3.3 Proposed System

It is important to find out the solution for this problem i.e. a solution must be found that solves the drawbacks of the PPM and the DPM strategy, which should be

- memory non-intensive,
- efficiently scalable,
- robust against packet pollution and
- independent of attack traffic patterns.

3.3.1 Traceback of DDoS by Entropy Variation Method

The packets that are passing through a router are categorized into flows, which are defined, by the upstream router where a packet came from, and the destination address of the packet. During non-attack periods, routers are required to observe and record entropy variations of local flows. Once a DDoS attack has been identified, the victim initiates a *pushback* process to identify the locations of zombies.

3.3.1.1 The Pushback Process

1. The victim first identifies which of its upstream routers are in the attack tree based on the *flow entropy variations* it has accumulated, and then
2. submits requests to the related immediate upstream routers.
3. The upstream routers identify where the attack flows came from based on their local entropy variations that they have monitored.
4. Once the immediate upstream routers have identified the attack flows,
5. they will forward the requests to their immediate upstream routers, respectively, to identify the attacker sources further.

This procedure is repeated in a parallel and distributed fashion until it reaches the attack source(s).

3.4 Advantages of the Proposed System

The proposed trace back mechanism possesses the following advantages:

- It *overcomes the inherited drawbacks of packet marking methods*, such as limited scalability, huge demands on storage space, and vulnerability to packet pollutions.
- It brings *no modifications on current routing software* because it works independently as an additional module on routers for monitoring and recording flow information.
- It will be effective for future packet flooding DDoS attacks because it is *independent of traffic patterns.*

It can *archive real-time trace back to attackers.* Once the short-term flow information is in place at routers, and the victim notices that it is under attack, it will start the trace back procedure.

3.5 Objectives of the Project

Following are the objectives of this project:

1. To find any DDoS attack that may be present in any network scenario.
2. To trace the origin of attack as well as ip address of the attacker with the help of Entropy variation technique as well as flow monitoring algorithm.

CHAPTER 4

DESIGN AND IMPLEMENTATION

4.1 A simple network with DDoS attack

In order to clearly describe the trace back mechanism, a sample network with DDoS attacks has been considered to demonstrate the trace back strategy. In a DDoS attack scenario as shown in Fig.5.1, the flows with destination as the victim include legitimate flows, such as f3, and a combination of attack flows and legitimate flows, such as f1 and f2. Compared with non-attack cases, the volumes of some flows increase significantly in a very short time period in DDoS attack cases. Observers at routers R1, R2 and Victim will notice the dramatic changes. Therefore, once the victim realizes an ongoing attack, it can push back to the LANs which caused the changes based on the information of flow entropy variations, and therefore, we can identify the locations of the attacker.

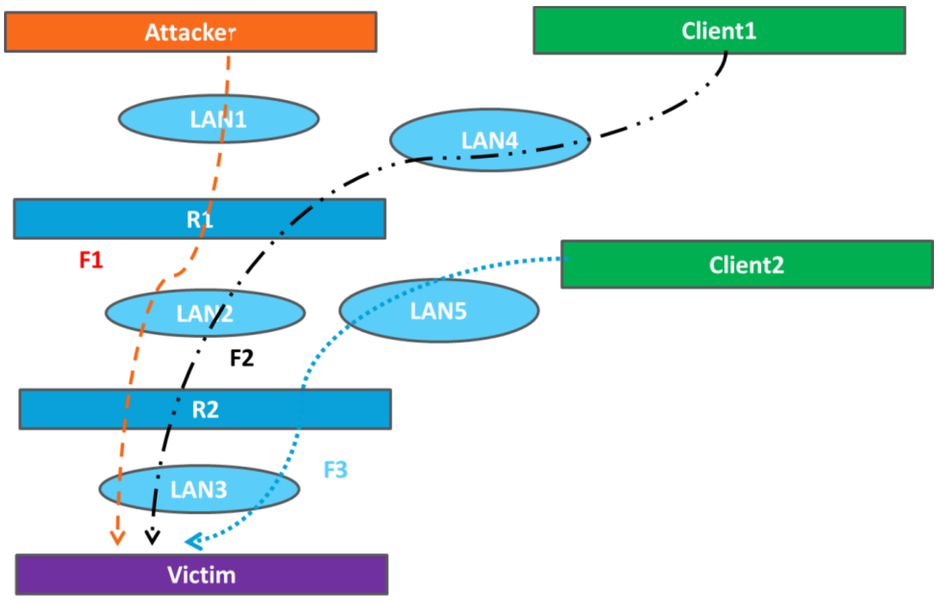


Fig 4.1 A simple network with DDoS attack

In the above-considered scenario two clients, one attacker, two routers and one victim is being considered. All these are connected through the LANs. Clients are the general legitimate users who are normally using the network. Attacker is the one who is responsible for the disruption of the normal operation of the Victim. Here during the design process two conditions has been considered i.e. the normal condition and the attacking condition. As the name itself says the normal condition defines the non-attacking condition in which the normal operation of the victim is not affected and the victim can do his job without any fail. However, the attacking condition is completely different from the normal condition. In the attacking condition the victim is considered as a target and hence attacker disrupts the normal operation by sending null packets or any unwanted packet. This results in low throughput of the victim and hence the performance in depleted.

The trace back can be done in a parallel and distributed fashion in this scheme. In Fig. 1, based on its knowledge of entropy variations, the victim knows attackers are somewhere behind router R2. Then the trace back request is delivered to router R2. Similar to the victim, router R2 knows that there is an attacker, who is somewhere behind the link to LAN2 .Then the trace back requests are further delivered to the edge routers of LAN1 andLAN5, respectively. Based on entropy variation information of router R1, it can infer that the attackers are located in the local area network, LAN1.The trace back request is hence passed to the upstream routers, until we locate the attackers in the appropriate LAN i.e. LAN1. Hence finally the attacker is found out.

4.2 System architecture

Here the packets are categorized which are passing through a router into flows. A flow is defined by a pair – the upstream router where the packet came from, and the destination address of the packet. Entropy is an information theoretical concept, which is a measure of randomness. The word entropy variation is defined to measure changes of randomness of flows at a router for a given time interval. The entropy variation is only one of the possible metrics. Statistical feature has been used; change point of flows, to identify the abnormality of DDoS attacks however, attackers could cheat this feature by increasing attack strength slowly. The other statistic metrics to measure the randomness can also be employed, such as standard variation or high order moments of flows. However the entropy variation was choose rather than others because of the low computing workload for entropy variations.

It is obvious that hunting down the attackers (zombies), and further to the hackers, is essential in solving the DDoS attack challenge. Hence it is very important to understand the proper working of the router. Generally, a router knows its local topology, e.g. its upstream routers, the local area network attached to the router, and the downstream routers. The router that is being investigated now is named as a local router. Here I as the set of positive integer, and R as the set of real number. We denote a flow on a local router as <u_i, d_j,t>, i,j € I, t € R where u_i is an upstream router of a local router R_i, d_j is the destination address of a group of packets which are passing through the local router R_i, and *t* is the current time stamp. For example, the local router R_i in Fig. 4.2 has two different incoming flows - the ones from the upstream routers R_j and R_k, respectively. C_i and C_k are the clients who needs the services of the victim's resources. Clients may be genuine users or may be attackers.

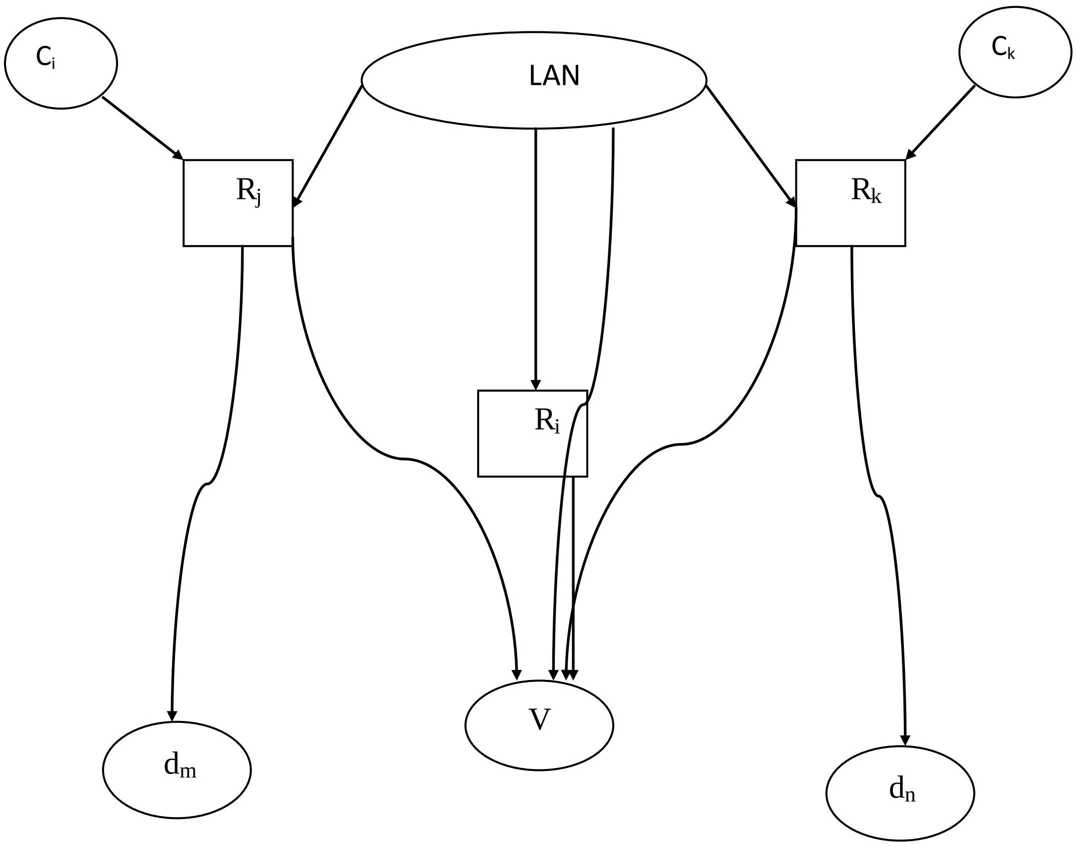


Fig 4.2 Network system architecture

CHAPTER 5

REQUIREMENTS

5.1 Hardware Requirements

PROCESSOR	: PENTIUM IV 2.6 GHz
RAM	: 512 MB DD RAM
MONITOR	: 15” COLOR
HARD DISK	: 20 GB

5.2 Software Requirements

FRONT END	: Java
TOOL USED	: JFrameBuilder
OPERATING SYSTEM	: Window’s Xp

CHAPTER 6

IMPLEMENTATION

6.1 Mathematical implementation

This kind of flows is named as transit flows. Another type of incoming flows of the local router R_i is generated at the local area network, these are called as local flows, and L is used to represent the local flows. All these incoming flows as termed as *input* flows, and all the flows leaving router R_i are named as output flows. The immediate upstream routers of the local router R_i is denote by u_i, $i \in I$, and set U as the set of incoming flows of router R_i. Therefore, $U = \{u_i, i \in I\} + \{L\}$. A set D $\{d_i, i \in I\}$ to represent the destinations of the packets which are passing through the local router R_i. If v is the victim router, then $v \in D$. Therefore, a flow at a local router can be defined as follows.

$$f_{ij}(u_{i,},d_j) = \{ < u_{i,},d_j,t > \mid u_{i,} \in U, d_j \in D, i, j \in I\} \qquad (1)$$

The $|f_{ij}(u_i,d_j,t)|$ is defined as the count number of packets of the flow f_{ij} at time t. For a given time interval ΔT , the variation of the number of packets for a given flow is defined as

$$N_{ij}(u_{i,},d_j,t + \Delta T) = |f_{ij}(u_{i,},d_j,t + \Delta T)| - |f_{ij}(u_{i,},d_j,t)| \qquad (2)$$

If $|f_{ij}(u_i,d_j,t)| = 0$ is used, then $N_{ij}(u_i,d_j,t+\Delta T)$ is the number of packets of flow f_{ij} which went through the local router during the time interval ΔT. Based on the large number theorem, the probability of each flow at a local router as follows:

$$P_{ij}(u_{i,},d_j) = \frac{N_{ij}(u_{i,}d_j)}{\sum_{i=1}^{\infty}\sum_{j=1}^{\infty} N_{ij}(u_{i,}d_j)} \qquad (3)$$

Where $P_{ij}(u_i,d_j)$ gives the probability of the flow f_{ij} over all the flows on the local router, Hence,

$$\sum_{i=1}^{\infty}\sum_{j=1}^{\infty} P_{ij}(u_{i,}d_j) = 1 \qquad (4)$$

Let F be the random variable of the number of flows during the time interval ΔT on a local router, therefore, the entropy of flows for the local router is defined as follows :

$$H(F) = - \sum_{i,j} P_{ij}(u_i, d_j) \log P_{ij}(u_i, d_j) \qquad (5)$$

In order to differentiate from the original definition of entropy, H(F) is called as entropy variation , which measures the variations of randomness of flows on a given local router.

6.2 Algorithms for implementation

6.2.1 Local Flow Monitoring Algorithm

1. Initialize the local threshold parameter, C, δ , and sampling interval Δ T.
2. Identify flows, $f_1, f_2, \ldots, f_n$,and set count number of each flow to zero.
3. When Δ T is over calculate the probability distribution and entropy variation as follows

   $P_i = x_i . (\sum_{i=1}^{n} x_i)^{-1}$ and $H(F) = - \sum_{i=1}^{n} p_i log\, p_i$
4. Save $x_1, x_2, \ldots, x_n$ and H(F).
5. If there is no dramatic change of the entropy variation H(F), namely, $|H(F)-C| \leq \delta$ Progress the mean $C[t] = \sum_{i=1}^{n} \alpha_i \, C\,[t-1]$, $\sum_{i=1}^{n} \alpha_i = 1$ and the standard variation $\delta[t] = \sum_{i=1}^{n} \beta_i \, \delta\,[t-1]$, $\sum_{i=1}^{n} \beta_i = 1$.
6. go to step 2.

Router1 and 2 and Victim have been coded to measure packet flows through them and mathematically compute the local flow probabilities (P_i) and Entropy variations H(F). Purpose of local flow monitoring algorithm is to capture frequent snapshots of local flow variations when there are no DDoS attack periods. From the data collected at specified time intervals , mean variation in entropy (Cmean) and threshold value for deviation (delta) are calculated and refined to ensure that any non-DDoS and legitimate packet flow measurements do not give rise to false positives. At Victim, we evaluate the Cmean and Delta from normalized entropy variations Hn= H(F)/H(F)Maximum = H(F)/Log_2 N where N= total number of flows through the system.

6.2.2 IP Traceback Algorithm

```
1. initialize a set A=Φ and obtain the local parameter of C and δ.
2. Let U = {u_i} , iϵ I be a set of the upstream routers D= {d_i }, iϵI be a set of destinations
   of the packets and v be the victim.
3. Define attack flows f_i = < u_j ,v > , i=1,2,….n, u_j ϵ U and sort the attack flows in the
   descent order and we have f_1’, f_2’, f_3’…… f_n’.
4. For i=1 to n { calculate H(F\ f_i’} and
       if |H(F)-C| > δ .
       then append the responding upstream router of f_i’ to set A .
       else break;
       end if;
       end for;
5. Submit the traceback requests to the routers in set A respectively and deliver the
   confirmed zombies information, set A, to the victim.
```

IP trace back algorithm is applied to write code in Routers 1 and 2 and Victim for detecting ,communicating and trace back for the DDoS attacker.

6.3 Components Implementation

In the course of understanding the trace back method, six components have been implemented that collectively makes the scenario of the DDoS attack.

6.3.1 Clients

As mentioned above clients are the normal users of the network. The process of implementation of the client is started with defining the arraylist in which the object of arraylist called pvtr is created. After the creation of the arraylist, A message will appear asking for the name of the file to be sent. With the help of the Scanner function the input will be read. Scanner is the complement of the formatter.

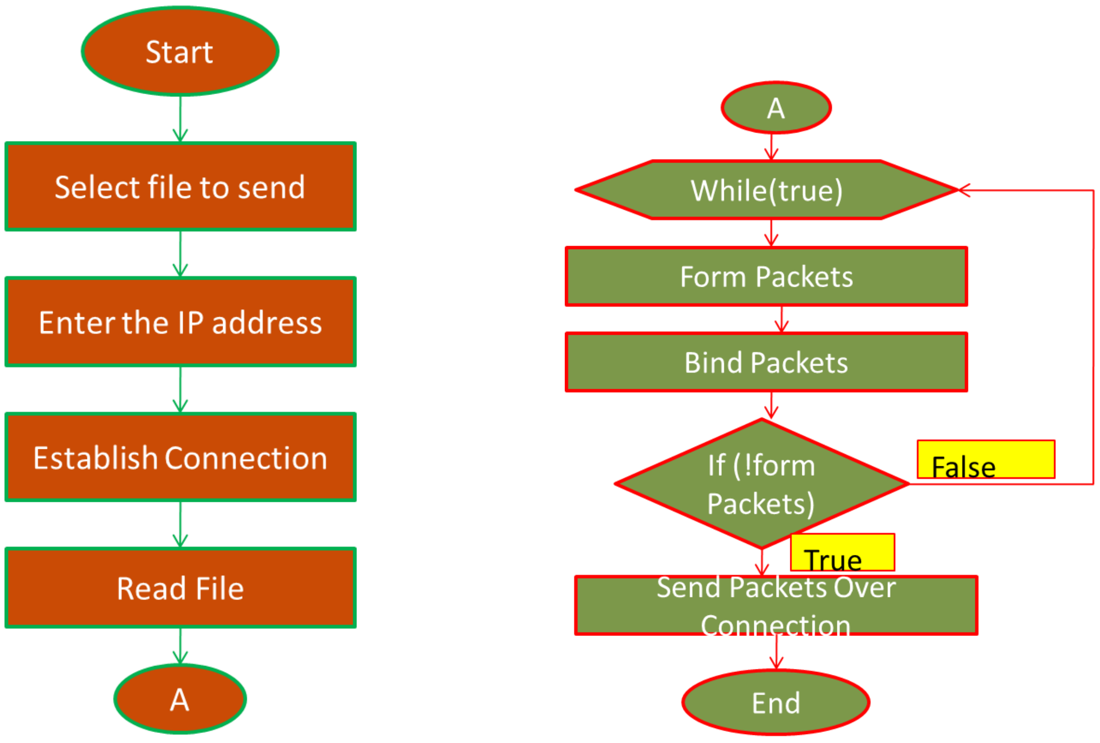


Fig 6.1 Client-component

As it can be easily understood from the flowchart that how the client works to get the job done. Firstly the filename is given as input as which file the client wants to send to the proper receiver. When the client finds the filename to be processed then it finds out the size of the file in total bytes. After finding the total bytes it stores into byte array. From the byte array, the packets are divided into 64 bytes a size packet, which is then packed into arraylist. From the array list each packet is read and sent through the socket to the particular destination. The need for storing packets into array list is that it is important to find out the total number of packets, which is to be processed.

A scanner reads formatted input and converts it into its binary form. Scanner can be used to read input from the console, a file, a string or any source that implements the Readable interface. After the name of the file to be sent is read, it will ask for the destination IP address. The IP address will be read and the packet will be sent to that address.

6.3.2 Attacker

As the name itself says, attacker is the one who is responsible for disrupting the normal operation of the victim. The attacker attacks the normal operation of the victim and is responsible for the degradation of the throughput. Here during the implementation process it has been considered that the packets what the attacker is sending are the null packets that are affecting the normal operation of the victim.

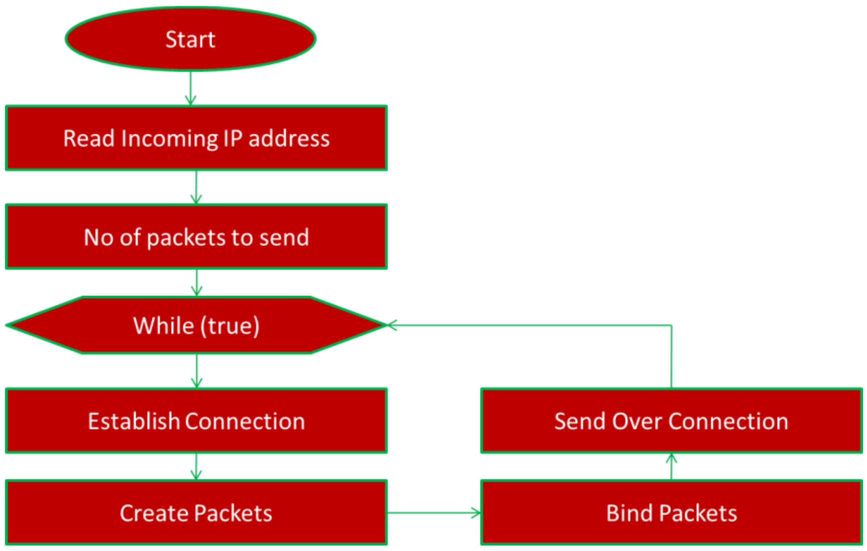


Fig 6.2 Attacker-component

From the above flowchart, the working of attacker can be easily understood. Firstly the attacker installs the program on the zombie systems. The zombie systems refer to the systems, which can easily be operated by the attacker. The attacker making use of that system attacks the victim and remains untraceable throughout the process. Firstly the attacker reads the destination IP address to which the attack is to be done. Then it defines the total number of packets to be sent to the victim so that the normal operation can be disrupted. After defining the total number of packets to be sent, it stores the null packets to the arraylist from which the packets are read and is sent to the appropriate destination.

6.3.3 Router

The router is the main part of the implementation process. The router is responsible for accepting the packets from one source and then forwarding it to the appropriate destination. Here first of all the router will wait for the socket connection. After the connection has been established, it will read the total no of packets to be received. Now this step is important in which prior information is given on how many packets will be received. After the total no of packets is known, it continues to receive packets and adds the packets to the arraylist. After the adding of the packets, it finds

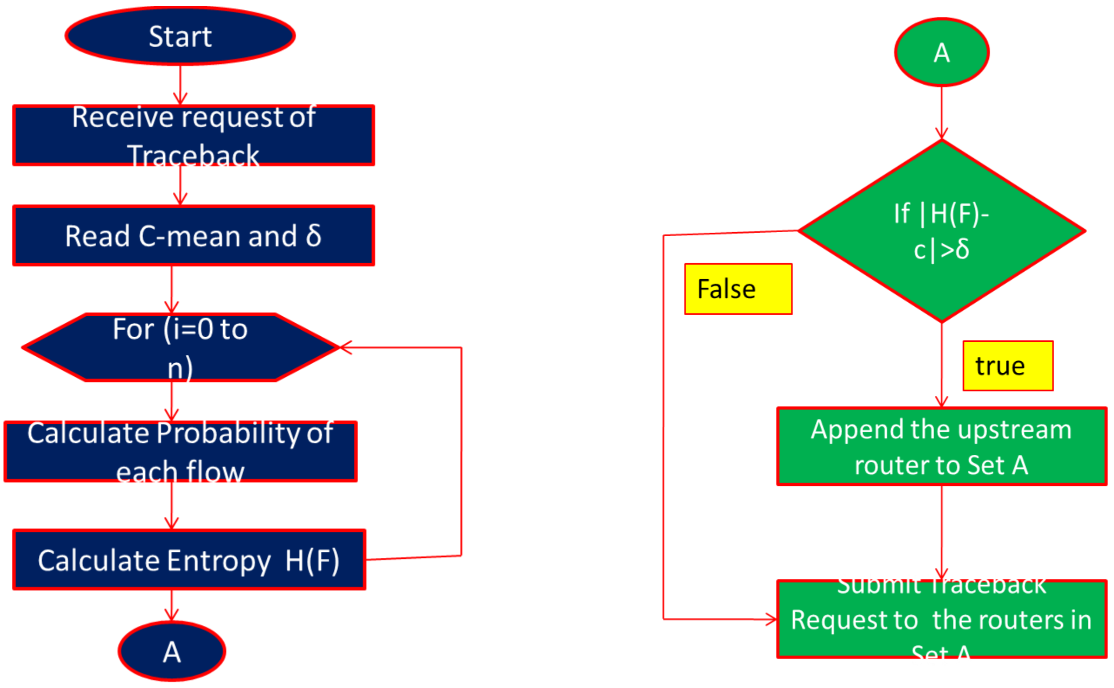

Start
Receive request of Traceback
Read C-mean and δ
For (i=0 to n)
Calculate Probability of each flow
Calculate Entropy H(F)
A
A
If |H(F)-c|>δ
False
true
Append the upstream router to Set A
Submit Traceback Request to the routers in Set A

Fig 6.3 Router while sending a file

from which flow the packet was coming. After the identification of the flow through which packet is coming, the count of that flow is incremented by one. The filename is given as input as which file the router needs to send to the proper receiver. When the router finds the filename to be processed then it finds out the size of the file in total bytes. After finding the total bytes it stores into byte array. From the byte array, the packets are divided into 64 bytes a size packet, which is then packed into arraylist. From the array list each packet is read and sent through the socket to the particular destination. The need for storing packets into array list is that it is important to find out the total number of packets, which is to be processed.

As the situation of attack arises in the abnormal condition, the router needs to implement the process of trace back. The process of trace back includes the concept of finding the source of the null packets or the unwanted packets, which is responsible for disruption of the normal working of the victim. First in the trace back condition a thread is created which waits for the socket from the victim or another source. Then a local parameter C that is mean and δ which is threshold is obtained and a set A is defined. Here the upstream and the downstream routers are considered. The upstream routers are the one, which lies along the previous routers through which the packets have passed and reached the destination while the downstream routers are vice-versa. Here three flows have been considered. Now with the help of the threshold, the probability distribution is calculated for each flow. With the help of calculated probability distribution, the entropy variation is calculated. Now if the difference between the entropy variation and the mean is greater than the threshold value then the responding upstream router is added to set A. After checking for all the flows the trace back request will be submitted to all the routers in set A. this process will be continued until the final culprit source will be found.

6.3.4. Victim

Victim receives flows from legitimate users(clients) as well as DDoS attacker.

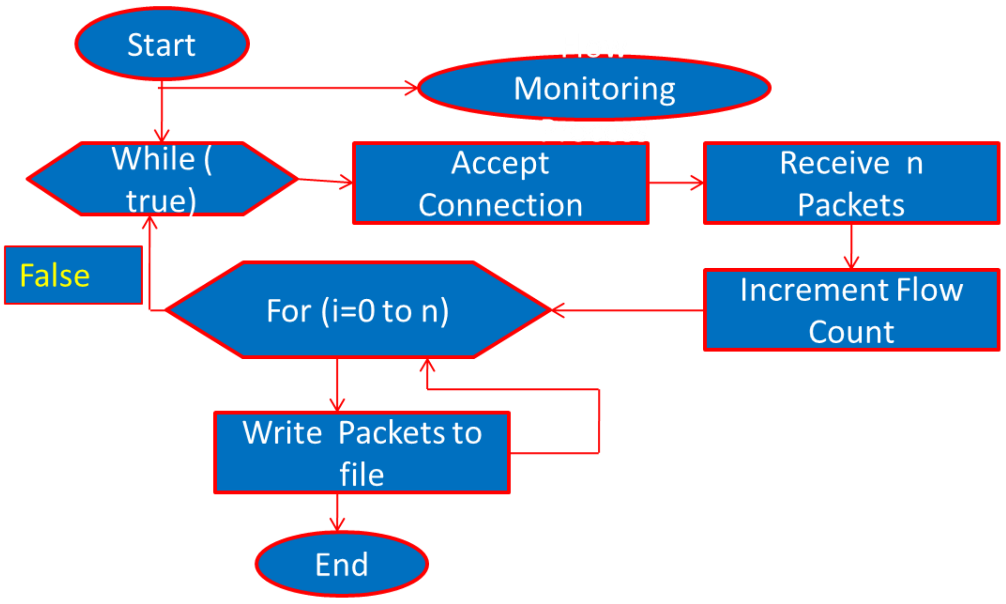


Fig.6.4 Victim-component

Victim is the one who is affected by the attack resulting in the degradation of the performance. The victim can be anywhere in the network and can be easily targeted by the attacker. The victim when attacked gives low output performance and the total throughput is degraded. The victim first accepts the socket connection and reads the total no of packets to receive. After it is prior known about the number of packets to be received, it began receiving the packets and increment the counter of that particular flow. After incrementing the count the packets are wrote into when there is any case of attack the victim needs to initiate the proper action towards the attack. Hence the victim needs to continuously monitor all the flows and hence finds and get to know if any unusual condition happens. The local flow-monitoring concept gives an idea of how to monitor all

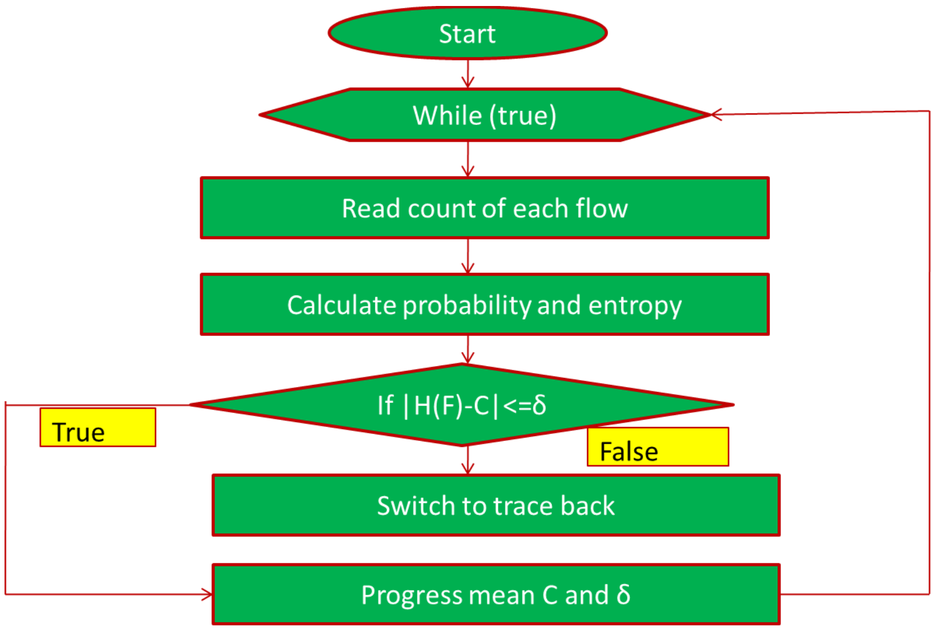


Fig 6.5 Flow monitoring at Victim

the flows. First of all there is need to initialize the parameter C,δ and the sampling time interval ΔT. The sampling time interval is the time interval up to which the flow will be monitored. For that particular time interval it will be checked that whether the attack has been made or not. After the initialization process there is need to identify the total flows through which the packets are coming to the victim. Here three flows have been considered and the count for each flow has been set to Zero. When that particular time interval ΔT is over then the probability distribution is calculated and hence the entropy variation is calculated. Now if there is any dramatic change in the entropy variation i.e. the difference between the C and the H(F) is greater than the threshold value then an attack is assumed. If the condition remains undisputed then the normal condition is assumed and the process of monitoring the flow is continued for the next time interval. Hence in this fashion the victim carries out the process of monitoring and checking the flows and hence defends against the attack. The local flow-monitoring algorithm is running at the nonattack period, accumulating information from normal network flows,and progressing the mean and the standard variation of flows. The progressing suspends when a DDoS attack is ongoing. Once a DDoS attack has been confirmed by any of the existing DDoS detection algorithms, then the victim starts the IP trace back algorithm.

6.4 Coding for implementation

Java networking API's include different packages, API's and methods which are used for implementation of this concept.

6.4.1 Using Server Socket

ServerSockets communicate using TCP connections. TCP guarantees delivery, so all there is need to do is have the applications read and write using a socket's InputStream and OutputStream.

6.4.2 Getting the IP address of a machine from its hostname

The InetAddress class is able to resolve IP addresses for you. Obtain an instance of InetAddress for the machine, and call the getHostAddress() method, which returns a string in the xxx.xxx.xxx.xxx address form.

```
InetAddress inet = InetAddress.getByName("www.davidreilly.com");

System.out.println ("IP: " + inet.getHostAddress ());
```

6.4.3 Performing a hostname lookup for an IP address

The InetAddress class contains a method that can return the domain name of an IP address. You need to obtain an InetAddress class, and then call its getHostName() method. This will return the hostname for that IP address. Depending on the platform, a partial or a fully qualified hostname may be returned.

```
        Initiatress inet = InetAddress.getByName ("209.204.220.121");

        System.out.println ("Host: " + inet.getHostName ());
```

6.4.4 Finding out the current IP address for my machine

The InetAddress has a static method called getLocalHost() which will return the current address of the local machine. You can then use the getHostAddress() method to get the IP address.

```
InetAddress local = InetAddress.getLocalHost();
System.out.println ("Local IP : " + local.getHostAddress());
```

6.5 Java Packages

The various java packages used in the project has been described below:

6.5.1 java.util.* package

Java Utility package is one of the most commonly used packages in the java program. The Utility Package of Java consists of the following components:

1. collections framework
2. legacy collection classes
3. event model
4. date and time facilities
5. internationalization

Here are some of the description of the utility classes of this package:

6.5.2 Data Structure Classes

Data Structure Classes are very useful classes for implementing standard computer science data structures: including BitSet, Dictionary, Hashtable, Stack and Vector. The Enumeration interface of java.util package is used to count through a set of values.

6.5.3 java.net.* package.

One of Java's strengths is simplified support for the development of network software. That support manifests itself through Java's Network API, a collection of classes and interfaces located in the packages java.net and javax.net. Java's Network API includes sockets .what the concept of a socket involves and what comprises a socket.

The Java Networking API (java.net) provides the interfaces/classes for the following functions:

- Addressing
- Making TCP connections
- Sending/Receiving Datagram Packets via UDP
- Locating/Identifying Network Resources
- Security

6.5.4 **java.io.* package**

The Java I/O package, a.k.a. java.io, provides a set of input streams and a set of output streams used to read and write data to files or other input and output sources. There are three categories of classes in java.io: input streams, output streams and everything else.

This chapter provide overviews of Java's I/O classes. They give information about what each class does and how you can use them. These pages do not provide any practical examples or details of each class. For more practical information regarding reading and writing data using these classes, see Input and Output Streams.

6.5.5 **Input Streams**

Input streams read data from an input source. An input source can be a file, a string, or memory-anything that can contain data. All input streams inherit from InputStream--an abstract class that defines the programming interface for all input streams. The InputStream class defines a programming interface for reading bytes or arrays of bytes, marking locations in the stream, skipping bytes of input, finding out the number of bytes that are available for reading, and resetting the current position within the stream. An input stream is automatically opened when you create it. You can explicitly close a stream with the close() method, or let it be closed implicitly when the object is garbage collected.

6.5.6 Output Streams

Output streams write data to an output source. Similar to input sources, an output source can be anything that can contain data: a file, a string, or memory.The OutputStream class is a sibling to InputStream and is used to write data that can then be read by an input stream. The OutputStream class defines a programming interface for writing bytes or arrays of bytes to the stream and flushing the stream. Like an input stream, an output stream is automatically opened when you create it. You can explicitly close an output stream with the close() method, or let it be closed implicitly when the object is garbage collected.

CHAPTER 7

RESULTS AND ANALYSIS

7.1 Testing

Software testing is the process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software item. Software testing is an activity that should be done throughout the whole development process. Software testing is one of the "verification and validation," or V&V, software practices.

There are two basic classes of software testing, *black box testing* and *white box testing.*

- *Black box testing* (also called functional testing) is testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions.
- *White box testing* (also called structural testing and glass box testing) is testing that takes into account the internal mechanism of a system or component.

7.1.1 Six Types of Testing

1. *Unit Testing :* Unit testing is the testing of individual hardware or software units or groups of related units Using *white box testing techniques,* testers (usually the developers creating the code implementation) verify that the code does what it is intended to do at a very low structural level.
2. *Integration Testing :* Integration test is testing in which software components, hardware components, or both are combined and tested to evaluate the interaction between them Using both *black and white box testing techniques*, the tester (still usually the software developer) verifies that units work together when they are integrated into a larger code base.
3. *Functional and System Testing:* Functional testing involves ensuring that the functionality specified in the requirement specification works. System testing involves putting the new program in many different environments to ensure the program works in typical customer environments with various versions and types of operating systems and/or applications. *System testing* is testing conducted on a complete, integrated system to evaluate the system compliance with its specified requirements. Using *black box testing techniques*, testers examine the high-level

design and the customer requirements specification to plan the test cases to ensure the code does what it is intended to do.

4. *Acceptance Testing* : Acceptance testing is formal testing conducted to determine whether or not a system satisfies its acceptance criteria (the criteria the system must satisfy to be accepted by a customer) and to enable the customer to determine whether or not to accept the system. This is done by the customer using *black box testing techniques*
5. *Regression Testing* : Regression testing is selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements. Since regression tests are run throughout the development cycle, there can be white box regression tests at the unit and integration levels and black box tests at the integration, function, system, and acceptance test levels.
6. *Beta Testing* : When an advanced partial or full version of a software package is available, the development organization can offer it free to one or more (and sometimes thousands) potential users or *beta testers*. These users install the software and use it as they wish, with the understanding that they will report any errors revealed during usage back to the development organization. These users are usually chosen because they are experienced users of prior versions or competitive products.

7.2 Testing Results

Unit testing, integration testing, functional and system testing and regression testing are conducted for this project as they are done by the software developers. Acceptance testing and Beta testing are to be done by the users and hence out of the scope for this academic project.

7.2.1 Unit Testing

ECLIPSE workbench is used to load and test the java code . *White box technique* is used to test individual code block for each component. Each code block has output print statements to verify the code for intended output. Eclipse console will print the output to verify the performance of the codes and loops in the blocks as per design.

7.2.2 Integration Testing

ECLIPSE workbench is used to test the integration of the components of Attacker, Clients, Routers and Victim using both *white and black box techniques*.

Integrity testing by *white box technique* is facilitated through the six component modules having code segments with appropriate *System.output.println( )* to display intended output on ECLIPSE console. Absence or wrong output is indicative of coding performance errors. Few cases of code blocks with such errors identified and rectified.

Integration testing through *black box technique* has been done by changing the inputs and validating the outputs as described in the table below.

Test case	**Expected output**	**Actual output**	**Test result**
File sent by Client with size 83392 bytes	File conversion to 64 bit packets of 1303 packets	1303 packets in router and victim display	Test passed
Attacker user input for Victims IP address 167.0.2.2 (non existing IP address)	Exception thrown on connection exception	`Console display` `Connection timed out: connect`	Test passed
Files sent to victim through both client 1 and 2 and attacker	Router display shall show attack flow as ***flow 1*** , client 1 flow as ***flow 2*** and client 2 flow as ***flow 3***	At router 1 ***flow 1*** and ***flow 2*** are displayed. At router 2 ***flow 1*** ***flow 2*** and ***flow 3*** are displayed.	Test passed

Table 7.1: Integration Testing

7.2.3 Functional and System testing

Functional and System testing are done by *black box testing technique*. Functional testing is done to verify the Entropy variation method application to DDoS attack scenario for realizing the objectives of

1. Detection of DDoS attack flow.
2. Identification of Attacker IP address.

Black box testing results are shown below

Test case	**Expected output**	**Actual output**	**Test result**
Attacker with IP address 192.168.0.100 sends empty packets in multiples of 50 no. to IP address 192.168.0.102	Entropy variation drops rapidly Attacker identified at Router 1 and msg flashed to victim with IP address of Attacker	Entropy variation drops rapidly below Threshold value 0.4 Msg flashed " Attacker identified" "Attacker IP is 192.168.0.100"	Test passed

Table 7.2: Functional Testing

System Testing is done for various testing inputs using black box testing techniques as follows

Test case	**Expected output**	**Actual output**	**Test result**
Non attack flows throw client 1 and 2 with out attacker by sending different file types such as PDF,JPEG,Excel and .Exe	No significant variation in entropy at victim and routers. No detection any DDoS attack	No significant change in entropy variation below threshold delta value of 0.4. No messages on DDoS attack	Test passed
Attack flows with varying multiples of packets (25, 50 and 100)	Detection time changes as per different intensities of attack	Detection times varied from 30 to 70 seconds depending on initial random flows through Victim	Test passed

Table 7.3: System Testing

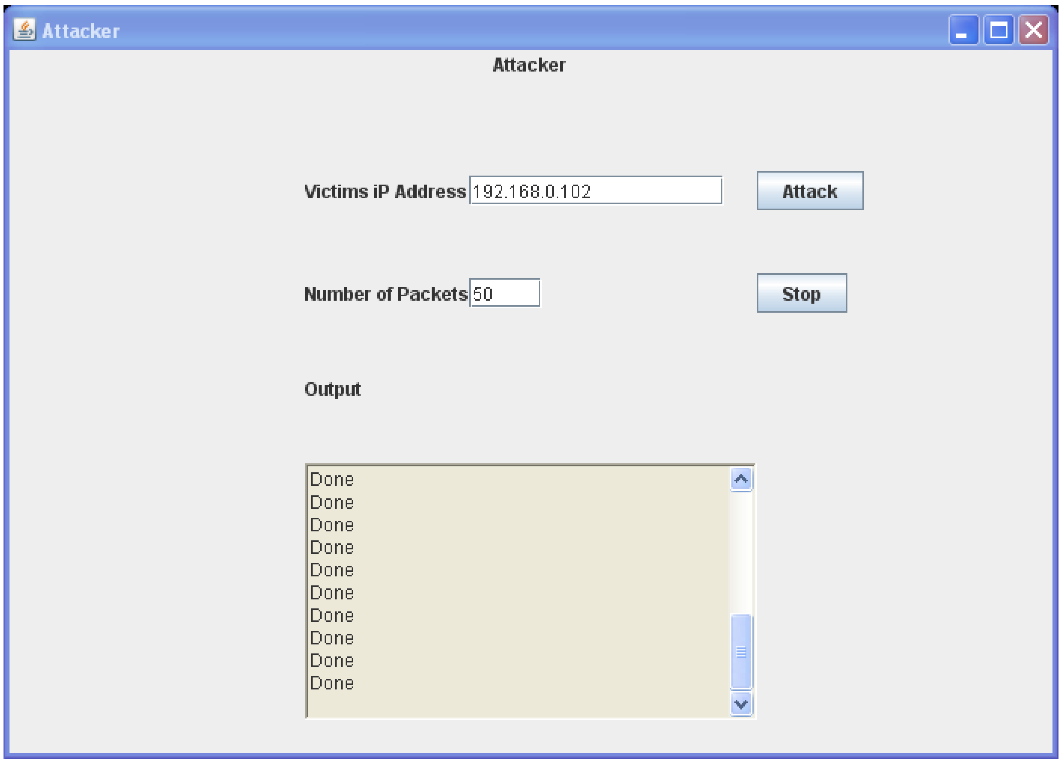
7.3 Results

Three systems are loaded with eclipse project with the finalized code. Networking between the three computers is done by WLAN facility in each of them. Eclipse project has been installed on each of the three LAPTOP computers. Tests have been carried out by making one of them attacker and other two as Victim and client. Project has been successful to demonstrate the Entropy variation method for DDoS Traceback

1. When no attack flow is received by the Victim there is very little variation in the local flow entropy.
2. Similarly when legitimate users client 1 and 2 send Packets to victim , the entropy variation does not exceed the threshold limits.
3. When a heavy attack flow is done by the Attacker , DDoS attack is successively detected by Victim, Router 2 and Router 1.
4. Router 1 identifies the attack flow ,obtains the IP address of Attacker and sends the message to Victim.
5. Identification of the attacker occurs within 45 to 60 seconds depending upon the local flow variations.

7.4 Snapshots

7.4.1 Attacker



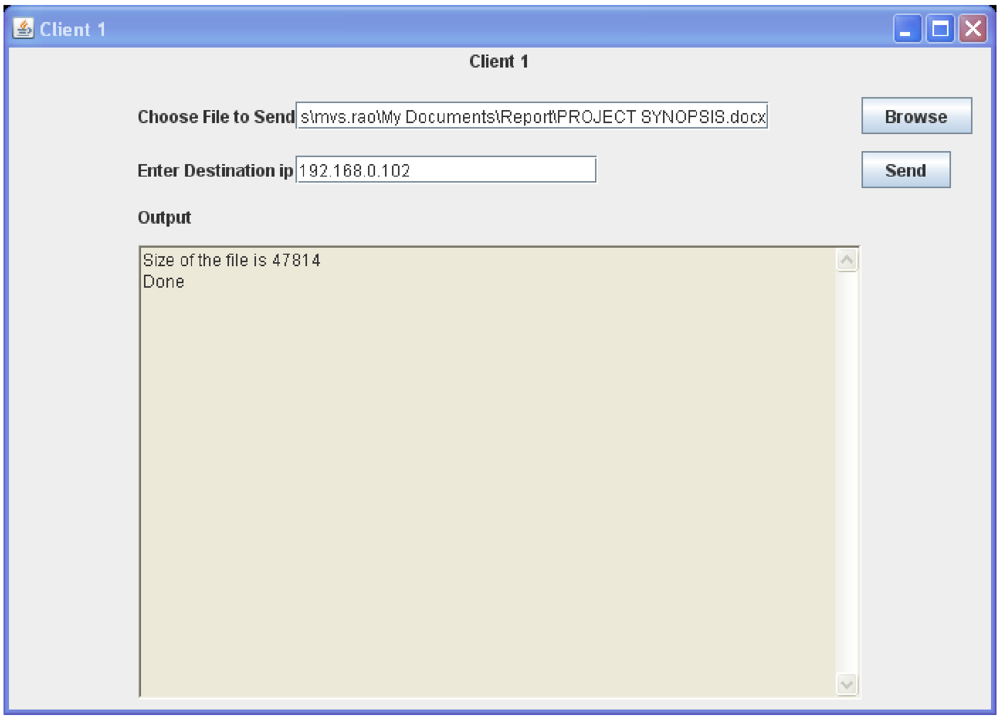
```
Victims iP Address 192.168.0.102
Number of Packets 50
```

```
Done
Done
Done
Done
Done
Done
Done
Done
Done
Done
```

Fig 7.4 Attacker-snapshot

Here IP address of victim is used to send the huge number of null packets to exhaust victim’s normal operation.

7.4.2 Client 1



s\mvs.rao\My Documents\Report\PROJECT SYNOPSIS.docx

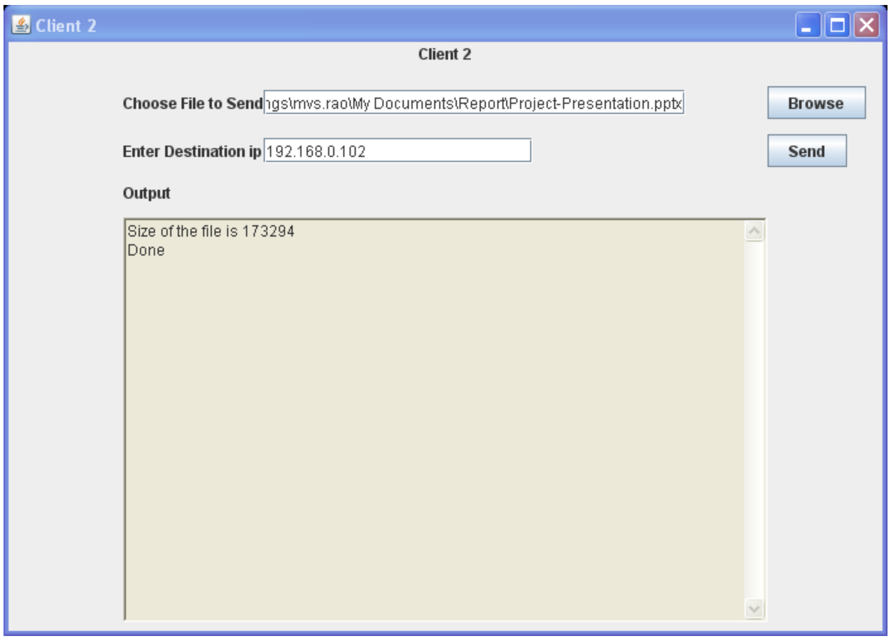
192.168.0.102

```
Size of the file is 47814
Done
```

Fig 7.5 Client 1-snapshot

Clients are the genuine users who uses the services provided by the network. Here a file has been attached and IP address of victim is entered to send those files to the victim. Client1 is the first client in the implemented network, which uses router1 and router2 to send files to the victim.

7.4.3 Client 2



ngs\mvs.rao\My Documents\Report\Project-Presentation.pptx

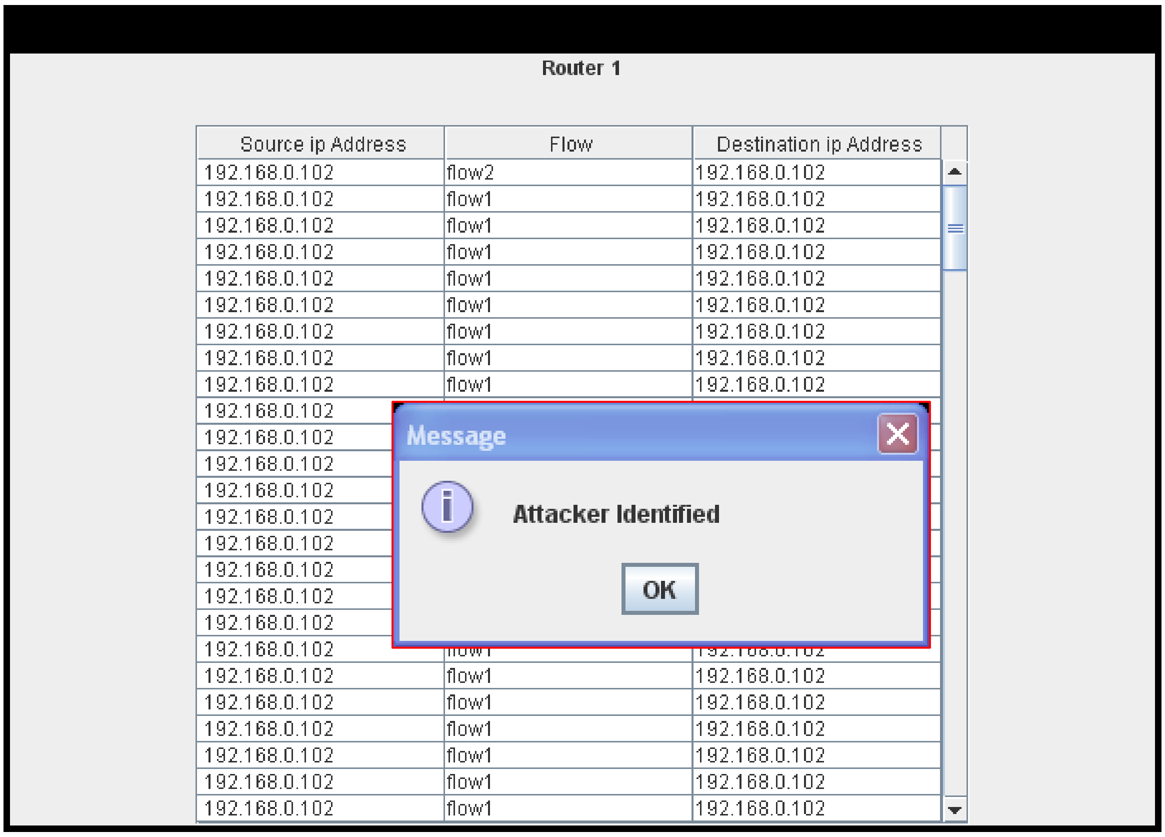
192.168.0.102

```
Size of the file is 173294
Done
```

Fig 7.6 Client 2-snapshot

Client2 is the second client in the implemented network, which uses router2 to send files to the victim.

7.4.4 Router 1



Router 1

Source ip Address	Flow	Destination ip Address
192.168.0.102	flow2	192.168.0.102
192.168.0.102	flow1	192.168.0.102
192.168.0.102	flow1	192.168.0.102
192.168.0.102	flow1	192.168.0.102
192.168.0.102	flow1	192.168.0.102
192.168.0.102	flow1	192.168.0.102
192.168.0.102	flow1	192.168.0.102
192.168.0.102	flow1	192.168.0.102
192.168.0.102	flow1	192.168.0.102
192.168.0.102		
192.168.0.102		
192.168.0.102		
192.168.0.102		
192.168.0.102		
192.168.0.102		
192.168.0.102		
192.168.0.102		
192.168.0.102		
192.168.0.102	flow1	192.168.0.102
192.168.0.102	flow1	192.168.0.102
192.168.0.102	flow1	192.168.0.102
192.168.0.102	flow1	192.168.0.102
192.168.0.102	flow1	192.168.0.102
192.168.0.102	flow1	192.168.0.102
192.168.0.102	flow1	192.168.0.102

Message

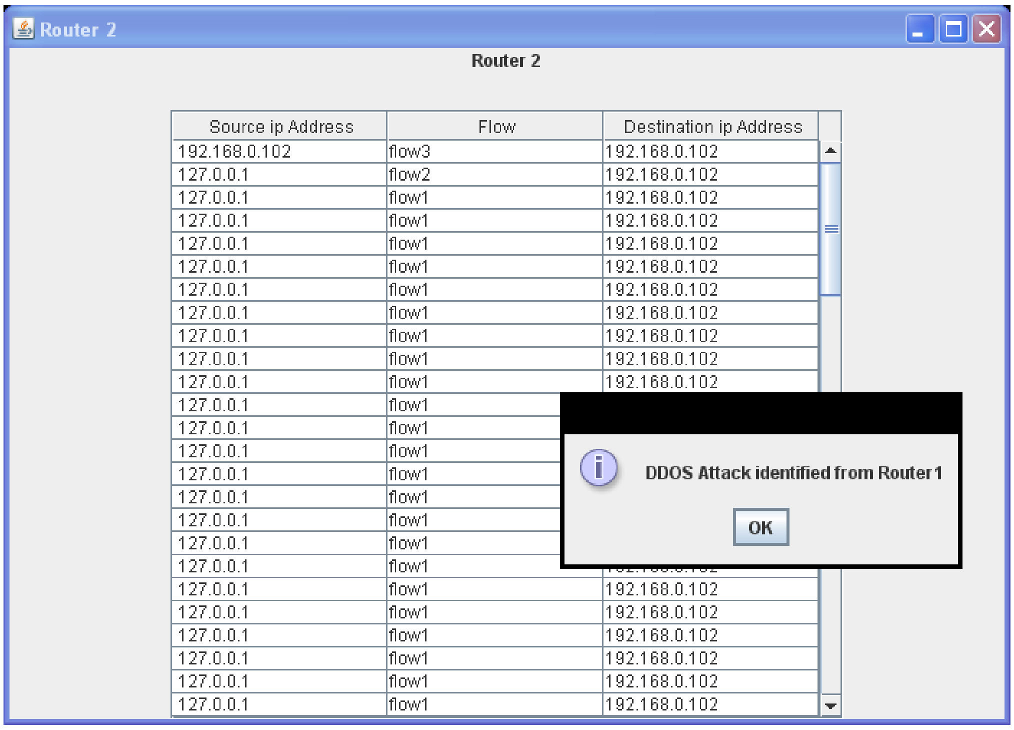
Attacker Identified

OK

Fig 7.7 Router 1-snapshot

Routers are responsible for forwarding the packets to the appropriate destination ip of victim and while trace backing it implements pushback operation to check the flows coming from upstream routers to identify the attacker. The above snapshot of router1 shows Source IP address from where packets are coming, flow number of the flow and the destination IP address where packets are being sent. Since router1 is directly connected to the attacker it also shows the alert box displaying “Attacker Identified” message.

7.4.5 Router 2


Router 2

Router 2

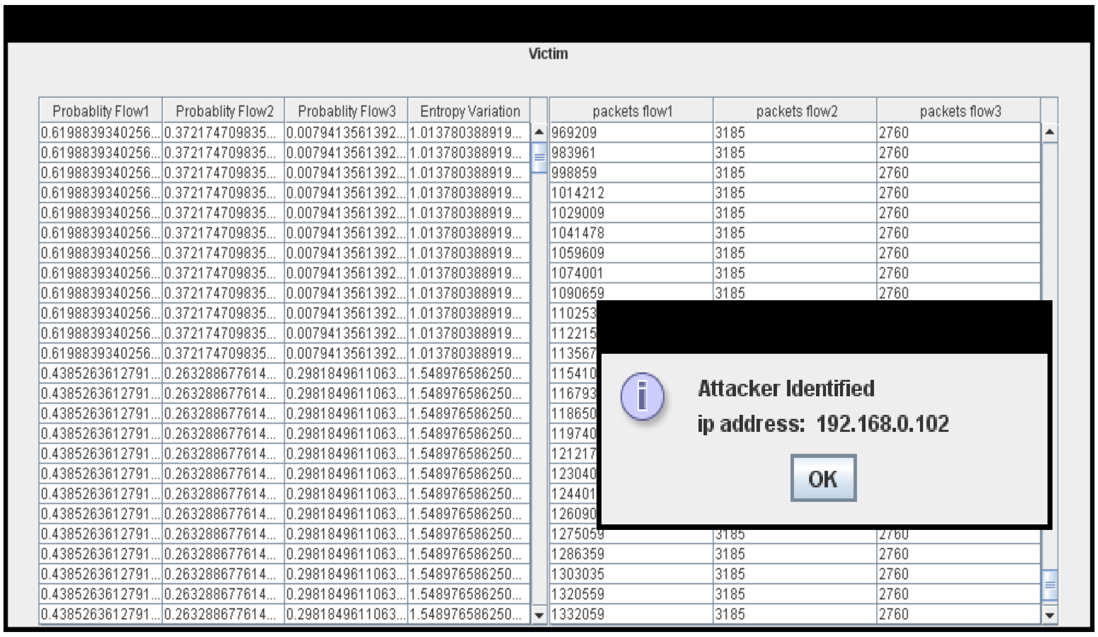
Source ip Address	Flow	Destination ip Address
192.168.0.102	flow3	192.168.0.102
127.0.0.1	flow2	192.168.0.102
127.0.0.1	flow1	192.168.0.102
127.0.0.1	flow1	192.168.0.102
127.0.0.1	flow1	192.168.0.102
127.0.0.1	flow1	192.168.0.102
127.0.0.1	flow1	192.168.0.102
127.0.0.1	flow1	192.168.0.102
127.0.0.1	flow1	192.168.0.102
127.0.0.1	flow1	192.168.0.102
127.0.0.1	flow1	192.168.0.102
127.0.0.1	flow1	
127.0.0.1	flow1	
127.0.0.1	flow1	
127.0.0.1	flow1	
127.0.0.1	flow1	
127.0.0.1	flow1	
127.0.0.1	flow1	
127.0.0.1	flow1	
127.0.0.1	flow1	192.168.0.102
127.0.0.1	flow1	192.168.0.102
127.0.0.1	flow1	192.168.0.102
127.0.0.1	flow1	192.168.0.102
127.0.0.1	flow1	192.168.0.102
127.0.0.1	flow1	192.168.0.102

DDOS Attack identified from Router1

Fig 7.8 Router 2-snapshot

The above snapshot of router2 also shows Source IP address from where packets are coming, flow number of the flow and the destination IP address where packets are being sent. Since router2 is not directly connected to the attacker it just implements pushback operation and send trace back request to upstream router.

7.4.6 Victim



Victim

Probablity Flow1	Probablity Flow2	Probablity Flow3	Entropy Variation
0.6198839340256...	0.372174709835...	0.0079413561392...	1.013780388919...
0.6198839340256...	0.372174709835...	0.0079413561392...	1.013780388919...
0.6198839340256...	0.372174709835...	0.0079413561392...	1.013780388919...
0.6198839340256...	0.372174709835...	0.0079413561392...	1.013780388919...
0.6198839340256...	0.372174709835...	0.0079413561392...	1.013780388919...
0.6198839340256...	0.372174709835...	0.0079413561392...	1.013780388919...
0.6198839340256...	0.372174709835...	0.0079413561392...	1.013780388919...
0.6198839340256...	0.372174709835...	0.0079413561392...	1.013780388919...
0.6198839340256...	0.372174709835...	0.0079413561392...	1.013780388919...
0.6198839340256...	0.372174709835...	0.0079413561392...	1.013780388919...
0.6198839340256...	0.372174709835...	0.0079413561392...	1.013780388919...
0.6198839340256...	0.372174709835...	0.0079413561392...	1.013780388919...
0.4385263612791...	0.263288677614...	0.2981849611063...	1.548976586250...
0.4385263612791...	0.263288677614...	0.2981849611063...	1.548976586250...
0.4385263612791...	0.263288677614...	0.2981849611063...	1.548976586250...
0.4385263612791...	0.263288677614...	0.2981849611063...	1.548976586250...
0.4385263612791...	0.263288677614...	0.2981849611063...	1.548976586250...
0.4385263612791...	0.263288677614...	0.2981849611063...	1.548976586250...
0.4385263612791...	0.263288677614...	0.2981849611063...	1.548976586250...
0.4385263612791...	0.263288677614...	0.2981849611063...	1.548976586250...
0.4385263612791...	0.263288677614...	0.2981849611063...	1.548976586250...
0.4385263612791...	0.263288677614...	0.2981849611063...	1.548976586250...
0.4385263612791...	0.263288677614...	0.2981849611063...	1.548976586250...
0.4385263612791...	0.263288677614...	0.2981849611063...	1.548976586250...
0.4385263612791...	0.263288677614...	0.2981849611063...	1.548976586250...

packets flow1	packets flow2	packets flow3
969209	3185	2760
983961	3185	2760
998859	3185	2760
1014212	3185	2760
1029009	3185	2760
1041478	3185	2760
1059609	3185	2760
1074001	3185	2760
1090659	3185	2760
110253		
112215		
113567		
115410		
116793		
118650		
119740		
121217		
123040		
124401		
126090		
1275059	3185	2760
1286359	3185	2760
1303035	3185	2760
1320559	3185	2760
1332059	3185	2760

Attacker Identified
ip address: 192.168.0.102

Fig 7.9 Victim-snapshot

Victim receives packets sent by the clients and monitors each flow. In the above snapshot probability distribution of each flow, their entropy variation and no of packets coming from each flow is maintained. Here the value of entropy variation has radically increased signifying an attack has been made. Hence it requests the trace back process to the router.

CHAPTER 8

CONCLUSION

. Our work proved that entropy variation method is not a mere theoretical application but practically capable of detection and trace back of DDoS attack in real time network environment. It has been shown that when attack flow enters the stream attack is identified and attacker IP address is traced back within 30 to 45 seconds of startup of DDoS attack. In the current work we have only limited number "flow" nodes(3) and hence we have calculated the important parameters of Cmean and delta using statistical methods outside the coding environment. However, for real networks having greater than 100 packet stream flows in the network it is possible to determine and update the Cmean and delta in dynamic live environment using neural network learning techniques.

One of the known drawbacks of any detection systems, is the risk of false positive detections and false negative detections.Compared to packet marking methods having inherent disadvantage of packet pollution defeating the detection mechanism, EVM(entropy variation method) is more flexible in reducing the chances of "false positives" and "false negatives" because it has provisions for (1).Changing the sensitivity of the detection system through continuous updating and adjustment of mean entropy variation (Cmean) and threshold value (delta) for detection of DDoS attack. This can be done dynamically through programming. (2) We can do extensive testing of the self-learning algorithms in real large networks under non-DDoS attack periods to eliminate the "false positives" and "false negatives" and (3)Speed of detection and trace back can also be adjusted through Cmean and Delta parameters but limited by the minimum "False positive" detections to be allowed for the concerned network systems

Future work could be carried out in the following promising directions:
Location estimation of attackers with partial information. When the attack strength is less than 7 times of the normal flow packet rate, this method cannot succeed at the moment. However, attack can be detected with the information that has been accumulated so far using traditional methods, e.g. the hidden Markov chain model, or recently developed tools, e.g. the network tomography. Differentiation of the DDoS attacks and flash crowds. In this project this issue has not been considered i.e. this trace back method may treat flash crowd as a DDoS attacks and therefore, resulting in false positive alarms.

BIBLIOGRAPHY

[1] ArborNetworks. "IP Flow-Based Technology," http:// www.arbornetworks.com.

[2] C. Patrikakis, M. Masikos, and O. Zouraraki, "Distributed Denial of Service Attacks," *The Internet Protocol Journal,* vol. 7, no. 4, pp. 13-35, 2004.

[3] 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, pp. Article 3, 2007.

[4] Y. Kim, W. C. Lau, M. C. Chuah *et al.*, "PacketScore: A Statistics-Based Packet Filtering Scheme against *On Dependable and Secure Computing,* vol. 3, no. 2, pp. 141-155, 2006.

[5] H. Wang, C. Jin, and K. G. Shin, "Defense Against Spoofed IP Traffic Using Hop- Count Filtering," *IEEE/ACM Transactions on Networking,* vol. 15, no. 1, pp. 40-53, 2007.

[6] Y. Chen, and K. Hwang, "Collaborative detection and filtering of shrew DDoS attacks using spectral analysis," *Journal of parallel and Distributed Computing,* vol. 66, pp. 137-1151, 2006.

[7] K. Lu, D. Wu, J. Fan *et al.*, "Robust and efficient detection of DDoS attacks for large- scale internet," *Computer Networks,* vol. 51, no. 9, pp. 5036-5056, 2007.

[8] R. R. Kompella, S. Singh, and G. Varghese, "On Scalable Attack Detection in the Network," *IEEE/ACM Transactions on Networking,* vol. 15, no. 1, pp. 14-25, 2007.

[9] P. E. Ayres, H. Sun, H. J. Chao *et al.*, "ALPi: A DDoS Defense System for High-Speed Networks," *IEEE Journal on Selected Areas in Communications,* vol. 24, no. 10, pp.1864-1876, 2006.

[10] R. Chen, J. Park, and R. Marchany, "A Divide-and- Conquer Strategy for Thwarting Distributed Denial-of- Service Attacks," *IEEE Transactions on Parallel and Distributed Systems,* vol. 18, no. 5, pp. 577-588, 2007.

[11] A. Yaar, A. Perrig, and D. Song, "StackPi: New Packet Marking and Filtering Mechanisms for DDoS and IP Spoofing Defense," *IEEE Journal of Selected Areas in Communication,* vol. 24, no. 10, 2006.

[12] A. Bremler-Bar, and H. Levy, "Spoofing Prevention Method," *IEEE INFOCOM, Miami, USA,* pp. 536- 547, 2005.

[13] J. Xu, and W. Lee, "Sustaining Availability of Web Services under Distributed Denial of Services Attacks," *IEEE Transactions on Computers*, vol. 52, no. 2, pp. 195- 208, 2003.

[14] W. Feng, E. Kaiser, and A. Luu, "Design and Implementation of Network Puzzles," *IEEE INFOCOM, Miami, USA*, pp. 2372-2382, 2005.

[15] X. Yang, D. Wetherall, and T. Anderson, "A DoS-Limiting Network Architecture," *ACM SIGCOMM*, pp. 241-252, 2005.

[16] Z. Duan, X. Yuan, and J. Chandrashekar, "Controlling IP Spoofing through Interdomain Packet Filters," *IEEE Transactions on Dependable and Secure Computing*, vol. 5, no. 1, pp. 22-36, 2007.

[17] F. Soldo, A. Markopoulou, and K. Argyraki, "Optimal Filtering of Source Address Prefixes: Models and Algorithms," *IEEE INFOCOM, Roi de Janeiro, Brazil*, 2009.

[18] A. El-Atawy, E. Al-Shaer, T. Tran *et al.*, "Adaptive Early Packet Filtering for Protecting Firewalls against DoS Attacks," *IEEE INFOCOM, Rio de Janeiro, Brazil*, 2009.

[19] T. Baba, and S. Matsuda, "Tracing Network Attacks to Their Sources," *IEEE Internet Computing*, vol. 6, no. 3, pp. 20-26, 2002.

[20] A. Belenky, and N. Ansari, "On IP Traceback," *IEEE Communication Magzine*, pp. 142-153, July, 2003.

[21] B. Al-Duwairi, and M. Govindarasu, "Novel Hybrid Schemes Employing Packet Marking and Logging for IP Traceback," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 5, pp. 403-418, May, 2006.

[22] M. T. Goodrich, "Probabilistic Packet Marking for Large- Scale IP Traceback," *IEEE/ACM Transactions on Networking*, vol. 16, no. 1, pp. 15-24, 2008.

[23] T. K. T. Law, J. C. S. Lui, and D. K. Y. Yau, "You Can Run, But You Can't Hide: An Effective Statistical Methodology to Traceback DDoS Attackers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 9, pp. 799-813, september, 2005.

[24] S. Savage, D. Wetherall, A. Karlin *et al.*, "Network Support for IP Traceback," *IEEE/ACM Transactions on Networking*, vol. 9, no. 3, pp. 226-237, June, 2001.

[25] A. Belenky, and N. Ansari, "IP Traceback With Deterministic Packet Marking," *IEEE Communications Letters*, vol. 7, no. 4, pp. 162-164, April 2003.

[26] D. Dean, M. Franlin, and A. Stubblefield, "An Algebraic Approach to IP Traceback," *ACM Transactions onInformation and System Security,* vol. 5, no. 2, pp. 119-137, May 2006.

[27] G. Jin, and J. Yang, "Deterministic Packet Marking based on Redundant Decomposition for IP Traceback," *IEEECommunications Letters,* vol. 10, no. 3, pp. 204-206, March 2006.

[28] Y. Xiang, W. Zhou, and M. Guo, "Flexible Deterministic Packet Marking: An IP Traceback System to Find the Real Source of Attacks," *IEEE Transactions on Parallel and Distributed Systems,* vol. 20, no. 4, pp. 567-580, 2009.

[29] C. Gong, and K. Sarac, "A More Practical Approach for Single-Packet IP Traceback Using Packet Logging and Marking," *IEEE Transactions on Parallel and Distributed Systems,* vol. 19, no. 10, pp. 1310-1324, 2008.

[30] K. Park, and H. Lee, "On the Effectiveness of Probabilistic Packet Marking for IP Traceback under Denial of Service Attack," *INFOCOM, Alaska, USA,* 2001.

[31] S.Yu, and W. Zhou, "Entropy-Based Collaborative Detection of DDoS Attacks on Community Networks," in the 6th Annual IEEE International Conference on Pervasive Computing and Communications, Hong Kong, 2008, pp. 566-571.

[32] S.Yu, W. Zhou, and R. Doss, "Information Theory Based Detection Against Network Behavior Mimicking DDoS Attacks," *IEEE Communications Letters,* vol. 12, no. 4, pp. 319-321, 2008.

[33] T. M. Cover, and J. A. Thomas, *Elements of Information Theory*: Wiley-Interscience, 2007.

[34] D. Moore, C. Shannon, D. J. Brown *et al.*, "Inferring Internet Denial-of-Service Activity," *ACM Transactions on Computer Systems*, vol. 24, no. 2, pp. 115-139, May 2006.

[35] J. Mirkovic, A. Hussan, S. Fahmy *et al.*, "Accurately Measuring Denial of Service in Simulation and Testbed Experiments," *IEEE Transactions on Dependable and Secure Computing,* vol. 6, no. 2, pp. 81-95, 2009.

[36] J. Mirkovic, P. Reiher, C. Papadopoulous *et al.*, "Testing a Collabotative DDoS Defense in a Red /Blue Team Exercise," *IEEE Transactions on Computers,* vol. 57, no. 8, pp. 1098-1112, 2008.

[37] H. Aljifri, "IP Traceback: A New Denial-of-Service Deterrent?," *IEEE Security & Privacy,* vol. 1, no. 3, pp. 24- 31, 2003.

[38] *Z. Gao*, and *N. Ansari*, "Tracing Cyber Attacks from the Practical Perspective," *IEEE Communications Letters*, vol. 43, no. 5, pp. 123-131, 2005.

[39] A. Yaar, A. Perrig, and D. Song, "FIT: Fast Internet Traceback," *IEEE INFOCOM, Miami, USA,* pp. 1395- 1406, 2005.

[40] A. C. Snoeren, C. Partiridge, L. A. Sanchez *et al.*, "Hash- Based IP Traceback," *AMC SIGCOMM*, 2001.

[41] A. C. Snoeren, C. Partiridge, L. A. Sanchez *et al.*, "Single- Packet IP Traceback," *IEEE/ACM Transactions on Networking*, vol. 10, no. 6, pp. 721-734, December, 2002.

[42] M. Sung, J. Xu, J. Li *et al.*, "Large-Scale IP Traceback in High-Speed Internet: Practical Techniques and Information-Theoretic Foundation," *IEEE/ACM Transactions on Networking*, vol. 16, no. 6, pp. 1253-1266, 2008.

[43] http://www.deakin.edu.au/noc.