Distributed Denial-of-Service Defense System

by

Lidiyawatie Hanasi (3821)

Dissertation submitted in partial fulfillment of
the requirements for the
Bachelor of Technology (Hons)
(Information CommunicationTechnology)

JUNE 2006

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Distributed Denial-of-Service Defense System

by

Lidiyawatie Hanasi

A project dissertation submitted to the
Information Communication Technology Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR OF TECHNOLOGY (Hons)
(INFORMATION COMMUNICATION TECHNOLOGY)

Approved by:

[illegible]

(Mr. Low Tan Jung)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

June 2006

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

[illegible]

LIDIYAWATIE HANASI

ABSTRACT

Distributed denial-of-service (DoS) attacks present a great threat to the Internet, and existing security mechanisms cannot detect or stop them successfully. The problem lies in the distributed nature of attacks, which engages the power of a vast number of coordinated hosts. To mitigate the impacts of DDoS attacks, it is important to develop such defenses system that can both detect and react against ongoing attacks. The attacks ideally should be stopped as close to the sources as possible, saving network resources and reducing congestion. The DDoS defense system that is deployed at the source-end should prevent the machines at associated network from participating in DDoS attacks. The primary objective of this project, which is developing a DDoS defense system, is to provide good service to a victim's legitimate clients during the attack, thus canceling the denial-of-service effect. The scope of study will cover the aspect of how the attack detection algorithms work and identify the attack traffic, hence develop appropriate attack responses. As a source-end defense against DDoS attacks, the attack flows can be stopped before they enter the Internet core and before they aggregate with other attack flows.

The methodology chosen for this project is the combination of sequential and iterative approaches of the software development process, which comprises of six main phases, which are initial planning phase, requirement definition phase, system design phase, coding and testing phase, implementation phase, and lastly maintenance and support phase. The system used a source router approach, in which the source router serves as a gateway between the source network containing some of the attack nodes and the rest of the Internet, to detect and limit DDoS streams long before they reach the target. This will be covered in the Findings section of the report. The Discussion section will be focus more on the architecture on the system, which having three important component; observation, rate-limiting and traffic-policing.

ACKNOWLEDGEMENT

First and foremost, I would like to express my hearties gratitude to ALLAH, The Al-Mighty, for giving me the strength and ability to carry out all the researches and tasks required for project completion. And also, I never would have completed on this project – DDoS Defense System without the help of many people. Thus, I would like to convey my highest appreciation to my project supervisor; Mr. Low Tan Jung for his time is consulting and facilitating me throughout my final year project. I also would like to thank the authors of the previous research papers whom greatly craft my mind in coming out with this thesis. For my closest friend; Nina, Elly and Sara, thank you for the time spent and supports, my other friends; thank you for giving such brilliant ideas. Moreover, lots of thanks to Mr. Ruslan Idris, for his help in helping and assisting me with his expertise and broad knowledge in network configuration throughout the project progress. Most of all, to my dearest family, thanks so much for the supports given to me. Without those mentioned above, this project never could be accomplished.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATION

CERT	Computer Emergency Response Team
DoS	Denial-of-Service
DDoS	Distributed Denial-of-Service
D-WARD	DDOS Network Attack Recognition and Defense
EMERALD	Event Monitoring. Enabling Responses to Anomalous Live Disturbances
FYP	Final Year Project
IBM	International Business Machines Corporation
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
ISP	Internet Service Provider
LRP	Linux Router Project
MULTOPS	A Data Structure for Bandwidth Attack Detection
PC	Personal Computer
RED	Random Early Detection
RFC	Request for Comments
SANS	SysAdmin, Audit, Networking, and Security
SDLC	Software Development Life Cycle
TCP/IP	Transmission Control Protocol/Internet Protocol
TCP/UDP	Transmission Control Protocol/User Datagram Protocol

CHAPTER 1
INTRODUCTION

1.1 BACKGROUND OF STUDY

1.1.1 Distributed Denial-of-Service Attacks

Denial of Service (DoS) attacks are particular interest and concern to the Internet community because they seek to render target system inoperable and/or target networks inaccessible. 'Traditional' DoS attacks, however, typically generate a large amount of traffic from a given host or subnet and it is possible for a site to detect such an attack in progress and defends themselves. Distributed denials of service (DDoS) attacks are a much more nefarious extension of DoS attacks because they are designed as a coordinated attack from many sources simultaneously against one or more targets.

DoS attacks under a number of guises have been around for decades, Distributed DoS attacks are much newer, first being seen in late June and early July of 1999. The first well-documented DDoS attack appears to have occurred in August 1999, when a DDoS tool called Trinoo was deployed in at least 227 systems, of which at least 114 were on Internet, to flood a single University of Minnesota computer; this system was knocked off the air for more than two days.

To describe and understand DDoS attacks, it is important to understand the terminology that is used to describe the attacks. DDoS attacks always involve a number of systems. A typical DDoS attack scenario might follow roughly the following steps:

1. An *intruder* finds one or more systems on the Internet that can be compromised and exploited (see figure below). This is generally accomplished using a stolen account on a system with a large number of users and/or inattentive administrators, preferably with a high-bandwidth connection to the Internet (many such systems can be found on college and university campuses).

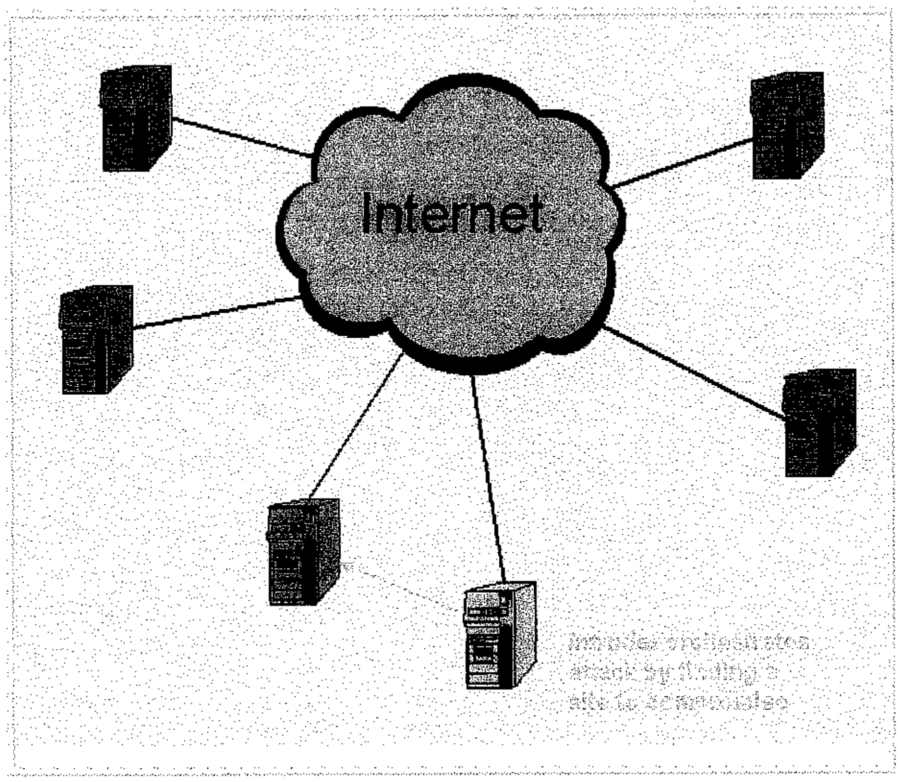


Figure 1.1: An intruder finds system on Internet than can be compromised.

2. The compromised system is loaded with any number of hacking and cracking tools such as scanners, exploit tools, operating system detectors, root kits, and DoS/DDoS programs. This system becomes the DDoS *master*. The master software allows it to find a number of other systems that can themselves be compromised and exploited. The attacker scans large ranges of IP network address blocks to find systems running services known to have security vulnerabilities. This *initial mass-intrusion phase* employs automated tools to remotely compromise several hundred to several thousand hosts, and installs DDoS agents on those systems. The automated tools to perform this compromise is *not* part of the DDoS toolkit but is exchanged within groups of criminal

hackers. These compromised systems are the initial victims of the DDoS attack. These subsequently exploited systems will be loaded with the DDoS *daemons* that carry out the actual attack (see figure below).

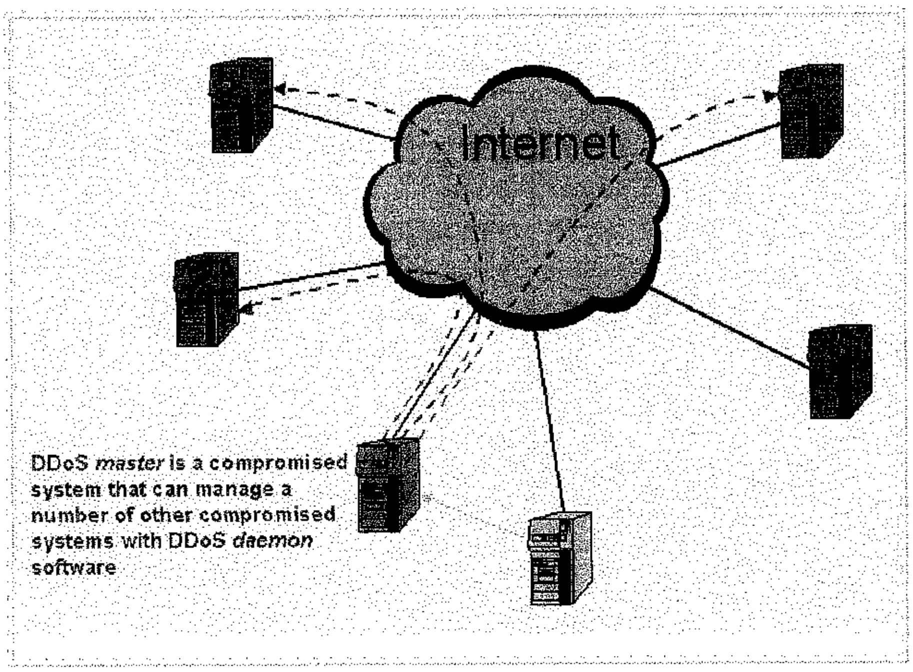

Internet
DDoS *master* is a compromised system that can manage a number of other compromised systems with DDoS *daemon* software

Figure 1.2: The exploited systems are loaded with the DDoS daemon.

3. The intruder maintains a list of *owned systems*, the compromised systems with the DDoS daemon. The actual *denial of service attack phase* occurs when the attacker runs a program at the master system that communicates with the DDoS daemons to launch the attack. Here is where the intended DDoS victim comes into the scenario (see next figure).

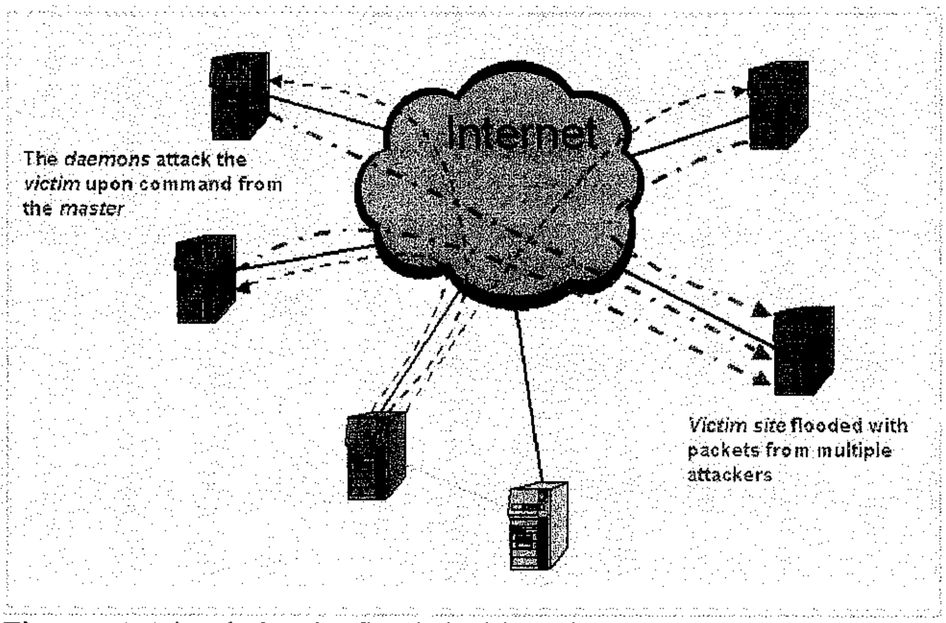

Internet
The *daemons* attack the *victim* upon command from the *master*
Victim site flooded with packets from multiple attackers

Figure 1.3: The victim site flooded with packets from multiple attackers.

Communication between the master and daemons can be obscured so that it becomes difficult to locate the master computer. Although some evidence may exist on one or more machines in the DDoS network regarding the location of the master, the daemons are normally automated so that it isn't necessary for an ongoing dialogue to take place between the master and the rest of the DDoS network. In fact, techniques are typically employed to deliberately camouflage the identity and location of the master within the DDoS network. These techniques make it difficult to analyze an attack while in progress and also to block attacking traffic and trace it back to its source.

To align those terms and the terms used by the hacker literature as well as early descriptions, we find the following synonyms:

- Intruder: Also called the *attacker* or *client*
- Master: Also called the *handler*
- Daemon: Also called an *agent*, *bcast (broadcast) program*, or *zombie*
- Victim: Always the victim

In addition, there are several features of DDoS attacks that severely challenge the design of successful defenses:

- **Use of IP source spoofing.**

  Attackers frequently use *source address spoofing* during the attack — they fake information in the IP source address field in attack packet headers. One benefit attackers receive from IP spoofing is that it is extremely difficult to trace the agent machines. This, in turn, brings several dire consequences. Since agent machines run a very low risk of being traced, information stored on them (i.e., access logs) cannot help to locate the attacker himself. This greatly encourages DDoS incidents. Furthermore, hiding the address of agent machines enables the attacker to reuse them for future attacks. Last, as attack packets carry a wide variety of addresses, they appear as if they come from many disparate sources; this defeats fair-sharing techniques that are a straightforward solution to resource overloading problems.

- **Large number of agent machines.**

  Even if traceback could be successfully performed in the face of IP spoofing, it is difficult to say what actions could be taken against hundreds or thousands of agent machines. Such a large number prevents any but crude automated responses aimed at stopping attack flows close to the sources.

- **Similarity of attack to legitimate traffic.**

  Any type of traffic can be used to perform a successful denial-of-service attack. Some traffic types require a higher attack volume for success than others, and attack packets of different types and contents target different resources. However, if the goal is simply to cripple the victim's operation, it can be met by sending sufficiently large volumes of any traffic and clogging the victim's network. Attackers tend to generate legitimate-like packets to perform the attack, obscuring the malicious flow within legitimate traffic. Since malicious packets do not stand out from legitimate ones, it is impossible to sieve legitimate from attack traffic based purely on examination of individual packets. A defense system must keep a volume of statistical data in order to extract transaction semantics from packet flows and thus differentiate some legitimate traffic (e.g. belonging to lengthy well-behaved transactions) from the attack traffic.

1.1.2 DDoS Defense

The seriousness of the DDoS problem and the increased frequency, sophistication and strength of attacks has led to the advent of numerous defense mechanisms. Yet, although it has been several years since the first distributed attacks were perpetrated, and many solutions have been developed since then, the problem is hardly dented, let alone solved.

The truth is that a site cannot defend itself from DDoS attacks alone. DDoS attacks depend upon the "community" of the Internet and defenses, therefore, depend upon the Internet community acting like a community with a common interest. And defending against attack includes ensuring that our own sites are not the source of attacks and our own networks do not forward attacks. The following will discuss some methods have been practiced by general users and professional experts that will help prevent the spread of DDoS attacks, by limiting the distribution of the tools and/or limiting the propagation of the offending "attack" packets

1.1.2.1 User and System Administrator Actions

Despite the best intentions and even the best distributed system management tools, the fact is that most computers today are largely managed by the local user. That means, in essence, that the local system has no security protections. However, whether the local host computer is a secretary's desktop system or the Web server for a company, there are procedures that can and should be taken to minimize the potential that an individual system will be compromised and itself attacked or used as a stepping-stone to attack others:

1. Keep abreast of the security vulnerabilities for all of the site's hardware, operating systems, and application and other software. This sounds like a Herculean task but it is essential to safeguarding the network. Apply patches and updates as soon as possible. Standardize on certain hardware, operating systems, and software where feasible to help manage the problem.

2. Consider using some form of "personal" firewall software to help detect an attack at particular systems.

3. Monitor the system periodically to test for known operating system vulnerabilities. Also periodically check to see what TCP/UDP ports are in use using the **netstat -a** command; every open port should be associated with a known application. All unused applications should be turned off.

4. Regularly monitor the system logs and look for suspicious activity.

5. The use of available tools to periodically audit the systems, particularly servers, to ensure that there have been no unauthorized/unknown changes to the file system, registry, user account database, etc.

6. Avoid downloading software from unknown, untrusted sites. If possible, know the author of the code. Even better is to download source code, review it (where feasible), and compile it on your system, if possible, rather than downloading binaries or executables.

7. Follow CERT, SANS, and TruSecure (ICSA) best practices procedures.

1.1.2.2 Local Network Actions

Even if a user locks down their system so that no vulnerability has gone unpatched and no exposure unprotected, the network itself (including the user community) can still be at risk. There are a number of procedures that local network managers and network administrators can take to protect all of their own users as well as the rest of the Internet community:

1. Every network connected to the Internet should perform egress address filtering at the router. *Egress filtering* means that the router should examine the IP Source Address field of every outgoing packet to the Internet to be sure that the NET_ID matches the NET_ID of the network. The user's firewall has historically been used to protect the user from attacks from the outside world. But those attacks come from somewhere so sites should also use the firewall to protect the outside world.

2. Networks should block incoming packets addressed to the broadcast address (the all-ones HOST_ID). There is no legitimate reason that an external network device should be sending a broadcast message to every host on your network.

3. To prevent the site from being used as a broadcast amplification point, turn off the Directed Broadcast capability at the router unless it is absolutely essential.

4. RFC 1918 defines a set of private IP addresses that are not to be routed on the Internet. These addresses include:

10.0.0.0/8	10.0.0.0-10.255.255.255	One Class A address
172.16.0.0/12	172.16.0.0-172.31.255.255	16 Class B addresses
192.168.0.0/16	192.168.0.0-192.168.255.255	256 Class C addresses

Table 1.1: Non routable IP addresses

5. In addition, there are a number of reserved IP addresses that are never assigned to "public" networks, including:

0.0.0.0/32	Historical broadcast address
127.0.0.0/8	Loopback
169.254.0.0/16	Link-local Networks
192.0.2.0/24	TEST-NET
224.0.0.0/4	Class D Multicast address range
240.0.0.0/5	Class E Experimental address range
248.0.0.0/5	Unallocated
255.255.255.255/32	Broadcast

Table 1.2: Reserved IP addresses

6. IP address spoofing is commonly employed by attackers and they commonly use one of the RFC 1918 private addresses or one of the other reserved addresses. Any packet that contains an RFC 1918 or reserved IP address in the IP Source Address or Destination Address field should be immediately discarded by the firewall and not ever sent to the Internet.

7. All unused application ports at the firewall should be blocked, particularly such ports as IRC (6665-6669/tcp) and those known to be associated with DDoS tools.

8. Use some form of intrusion detection system (IDS) to protect the network. Considering providing every system with "personal" firewall software to help detect an attack at individual systems; this is particularly potentially useful at sites (such as colleges) that have a large number of systems in front of a firewall (it is no coincidence that so many daemons reside on college and university computers that have been "owned").

9. Network activity should be regularly monitored so that aberrations in traffic flow can be quickly detected.

10. Educate the users about things to watch for on their systems and how to report any irregularity that might indicate that someone or something has tampered with their system. Educate help desk and technical support to assist those users who make such reports. Have an intelligence gathering system within your organization so that such reports are centrally known so that trends can be spotted and responses devised.

11. Follow CERT, SANS, and TruSecure (ICSA) best practices procedures.

1.1.2.3 ISP Actions

The Internet Service Providers (ISPs) offer the last hope in defeating the spread of a DDoS attack. While the ISP cannot take responsibility for locking down every customer's host systems, the ISPs have -- and should accept -- the responsibility to ensure that their network does not carry packets that contain obviously "bad" packets. Some of the procedures taken by ISPs include:

1. As mentioned above, IP address spoofing is commonly employed by attackers using one of the RFC 1918 private addresses or one of the other reserved addresses. Amazingly, many ISPs will route these packets. Indeed, there is no entry in their routing table telling them where to send the packets; they merely forward them to a default upstream ISP. Any packet that contains any RFC 1918 or reserved IP address in the IP Source Address or Destination Address field should be immediately discarded.

2. Perform ingress (and egress) address filtering. *Ingress filtering* means that they should examine every incoming packet to their network from a customer's site and examine the IP Source Address field to be sure that the NET_ID matches the NET_ID assigned to that customer. This will require additional configuration at the router and may even result is slight performance degradation but the tradeoff is certainly well worth the effort. The ISPs should

also perform egress filtering to check their outbound packets to upstream and peer ISPs.

3. IP directed broadcasts should be disabled.
4. Educate customers about security and work with them to help protect themselves.

Most of the ISP communities take at least some of these steps. Users should insist that their ISPs provide at least these protections and should not do business with those who don't. in addition, the TruSecure (formerly ICSA) ISP Security (ISPSec) community is a good source of information for ISPs.

1.1.2.4 Others DDoS Defense Tools

Responses to DDoS attacks are not limited to the defensive steps listed above. Indeed, proactive responses to the prevention and detection of DDoS attacks is an active area of research. Additionally, there are numerous approaches to DDoS defense and network security in general. Thus, in the following sections provided detailed descriptions of related DDoS defense approaches, which are consisting of victim-end defenses, source-end defenses, and distributed defenses. As depicted on Figure 1.4, each of the involved networks (source, intermediate, or victim) can host DDoS defense systems.

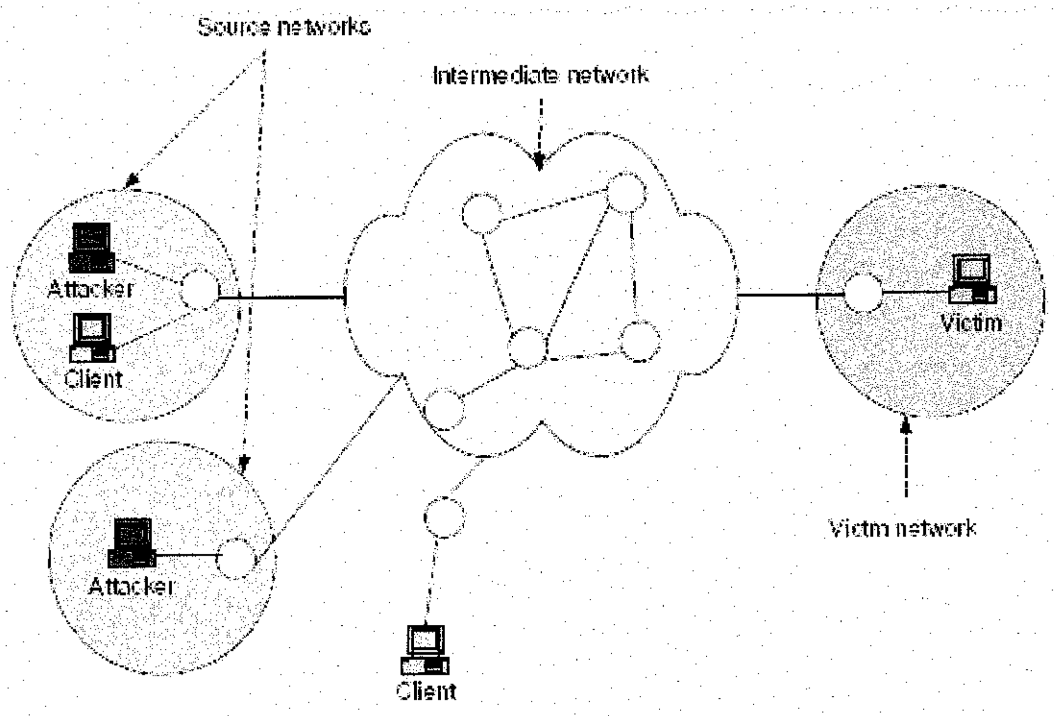


Figure 1.4: Point of Defense

- **Victim-End Defenses**

Historically, the majority of DDoS defense systems have been designed for victim-end deployment. This is understandable since the victim suffers the largest damage from a DDoS attack and is therefore motivated to invest in a defense system. A victim-end DDoS defense system facilitates easy detection because it can closely observe the victim, model its behavior and notice any anomalies.

However, the range of response is limited. The defense system is on the path of the full-force attack, and may be overwhelmed by a large traffic volume. The point of failure is then simply moved from the target to the DDoS defense system. Alternately, the attacker may send enough traffic to overwhelm the victim's network connection in front of the defense system. The point of DDoS is then beyond the defense system's scope. The other consequence of a large traffic volume is the limited amount of processing and storage that defense system can commit to. The differentiation of legitimate streams from attack streams is complex at this point, since they have been heavily aggregated by the time they reach the victim network. To perform sophisticated traffic profiling a system needs a large amount of storage and computational power to store and examine statistics on each stream. In the presence of IP spoofing, this is infeasible as each packet will appear to come from a different source. Poor traffic separation, in turn, leads to large collateral damage during a response.

- **Intermediate-Network Defenses**

The danger of a DDoS attack on network resources that is still present in victim-end defense was addressed by moving the defense further upstream, into the intermediate network. An intermediate-network defense system, usually installed at a core router, detects the attack through anomalies observed at this router. As core routers handle large-volume, highly aggregated traffic, they are likely to overlook all but large-scale attacks. Victim resources are frequently severely depleted by attacks that look like small glitches in the busy buffer of a core router. Detected attacks can be quickly suppressed, thanks to abundant network resources. However, response is likely to inflict collateral damage as core routers can only accommodate simple rate-limiting requests and cannot dedicate memory or processor cycles to traffic profiling.

- **Source-End Defenses**

As DDoS defense is pushed further from the victim to the source, detection capability diminishes. A source-end defense system can no longer easily observe the effect of incoming traffic on the victim. Further, as it may monitor only a small portion of the attack, the defense system has difficulties in detecting anomalies. On the other hand, response effectiveness increases with proximity to the sources. A small attack volume enables an effective response as it is unlikely to overwhelm the defense system. The small volume and degree of aggregation also facilitates complex profiling that, in turn, minimizes collateral damage.

- **Distributed Defenses**

Distributed systems for DDoS defense combine actions of victim-end, source-end and sometimes of intermediate-network defense systems. Victim-end defenses detect the attack and deliver the alert to other participants, who then cooperate to suppress attack streams. The goal is to install responses as close to the sources as possible, thus minimizing collateral damage.

Distributed defenses are likely to be the proper solution for handling the DDoS threat. However, they are infrastructural solutions—they span multiple networks and administrative domains and represent major undertakings of many Internet participants. Such systems are difficult to deploy and maintain. Further, the required cooperation of defenses is hard to achieve due to distributed Internet management and strictly autonomous operation of administrative domains. Securing and authenticating the communication channels also incurs a high cost if the number of participants is large.

1.2 PROBLEM STATEMENT

Distributed denial-of-service attacks present a great threat to the Internet, and existing security mechanisms cannot detect or stop them successfully. The problem lies in the distributed nature of attacks, which engages the power of a vast number of coordinated hosts. The response to the attack needs to be distributed also, but cooperation between administrative domains is hard to achieve, and security and authentication of participants incur a very high cost.

In the absence of effective defense mechanisms, the DDoS effect lasts for the entire duration of the attack (i.e., as long as key resources are being tied with malicious traffic), and vanishes quickly once the attack is aborted. Since machine resources are usually shared among many applications, the DDoS effect inflicts significant damage, not only on client transactions with the victim, but on the victim's total operation. The victim experiences a significant slowdown in all applications sharing the targeted resource, and frequently also connectivity disruption.

In the DDoS attacks, the attacking packets come from tens or hundreds of addresses rather than just one, as in a 'traditional' DoS attacks. Any DoS defense system that is based upon monitoring the volume of packets coming from a single address or single network will then fail since the attacks come from all over. Rather than receiving, for example, a thousands gigantic Pings per second from an attacking site, the victim might receive one Ping per second from 1000 attacking sites.

In addition, the cooperation of distributed sources makes DDoS attacks hard to combat or trace back. Any defense against these attacks must address their distributed nature to be successful. Several DDoS defense systems have been proposed that deploy cooperation between intermediate routers to combat the attacks. These routers are augmented to monitor traffic and grant requests for rate-limiting of the systems they deliver to their peers. While these approaches seem promising, they require the cooperation of every router from the source to the destination host for response propagation. Since the Internet is not under any single administrative control, global deployment of such mechanisms is hard to enforce. Also, cooperation between routers requires strongly secured and authenticated communication, which incurs high cost.

Since there are many attack variations and many dimensions in which attacks can still evolve while preserving the ability to inflict damage on the victim, this feature makes it very challenging to design successful defenses systems. Due to attack variety, defense system must maintain a volume of statistical data in order to detect attacks and sieve legitimate from attack traffic. This, however, incurs high operation cost. On the other hand, attackers can easily bypass or trick defenses with slight modifications to their attacks. Any such modifications require added complexity in defense mechanisms (in order to handle the new attack class), thus skyrocketing the cost.

1.3 OBJECTIVES AND SCOPE OF STUDY

The primary objective of this project, which is developing a DDoS defense system, is to provide good service to a victim's legitimate clients during the attack, thus canceling the denial-of-service effect. Ideally, clients should perceive little or no service degradation while the attack is ongoing. The secondary objective is that to alleviate the effect of the attack on the victim so that its resources can be dedicated to legitimate clients or preserved.

The scope of study will cover the aspect of how the attack detection algorithms work and identify the attack traffic, hence develop appropriate attach responses. In this project, it will be focusing on a specified defense approach, with specified attack responses, as well as a specified points of defense will be deployed as explained below:

- **Defense Approach**

This project will be focusing in ***Responsive Approaches*** to detect the occurrence of the attack and respond to it. In order to be successful, response approaches must meet following requirement:

1. ***Accurate detection***

   The system should be able to detect all attacks that inflict damage at the victim.

2. ***Effective response***

   The system must stop the attack flows, regardless of their volume or distribution. Alternately, in the case of response by agent identification, the system must be able to accurately identify the majority of attack machines regardless of their distribution. This identification must be prompt so that the action can be taken while the attack is on-going. Ideally, identification responses should identify not only the agent machines, but also the master and the attacker machines.

3. ***Selective response***

   The system must differentiate between legitimate and attack packets, and ensure good service to legitimate traffic during the attack. Collateral damage due to the response must be lower than the damage suffered by legitimate clients in the absence of response. This requirement does not pertain to agent identification approaches.

- **Attack Responses**

  There are two categories of attack responses, which are ***destination filtering*** (for attack victims) and ***source filtering*** (for attack sources). However, since this project will be kind of source-end base defense system, therefore the attack response will be the source filtering. Defenses in this category monitor the network traffic sent from some source networks, and mitigate the impacts of ongoing attack traffic originating from these sources.

- **Points of Defense**

  A DDoS defense system can either be deployed as an autonomous (single point) system or as a distributed system. Autonomous systems consist of a single defense node that observers the attack and applies the response. Distributed systems consist of multiple defense nodes (frequently with same functionality) that are deployed at various locations and organized into a network. In as autonomous defense, a DDoS attack streams originate from distributed attack machines are forwarded by core routers and converge at the victim network or some nearby core router. The process is observed according to the interaction of three types of network: ***source networks*** that unwittingly host attack machines, several ***intermediate networks*** that forward attack traffic to the victim, and the ***victim networks*** that host the target. For this particular project, it will be focusing on the source network for autonomous based system.

  Source-End Defense: Placing DDoS defense close to the sources of the attack has many advantages over placing it further downstream. The attack flows can be stopped before they enter the Internet core and before they aggregate with other attack flows, and achieve the power to create network congestion and exhaust resources of the victim and intermediate routers. Being close to the sources can facilitate easier traceback and investigation of the attack. Due to the low degree of flow aggregation, more complex detection strategies can be deployed to achieve higher accuracy. Also, routers closer to the sources are likely to relay less traffic than core routers and can dedicate more of their resources to DDoS defense at a lower performance impact.

CHAPTER 2
LITERATURE REVIEW AND THEORY

We've already familiar with denial-of-service (DoS) attacks for few decades, however, distributed denial-of-service (DDoS) is new. The main difference between DoS and DDoS attacks is in scale – DoS attacks use one attack machine (to generate malicious traffic) while DDoS attacks use large numbers of attack machines [1]. The scale difference also invokes differences in operation modes.

The large number of attack machines allows DDoS attackers certain recklessness, means that they frequently trade sophistication for brute force, using simple attack strategies and packet contents to overload victim resources. The lack of effective defense mechanisms, even for simple attacks, offers no motivation for attackers to design more sophisticated ones. Once defenses successfully counter one attack class (e.g., like ingress filtering has countered random IP source spoofing), attackers quickly deployed slight modifications in their attacks to bypass defensive actions [2]. However, any such modifications require added complexity in defense mechanisms in order to handle the new attack class.

This leads many researchers to come with new defense mechanisms, including source router approach. Placing the DDoS defense close to the sources of the attack has many advantages over placing it further downstream [3]. The attack flows can be stopped before they enter the Internet core and before they aggregate with other attack flows, and achieve the power to create network congestion and exhaust resources of the victim and intermediate routers. For example, the D-DWARD [4] system is installed at the source router that serves as a gateway between deploying network (source network) and the rest of the Internet.

2.1 ATTACKER GOALS

The goal of a DDoS attack is to inflict damage on the victim. Frequently the ulterior motives are personal reasons (a significant number of DDoS attacks are perpetrated against home computers, presumably for purposes of revenge), or prestige (successful attacks on popular Web servers gain the respect of the hacker community). However, it is not unlikely that some DDoS attacks are performed for material gain (damaging competitor's resources, such as a case of Linux fans attacking SCO [5] because of its lawsuit against IBM) or for political reasons (a country at war could perpetrate attacks against its enemy's critical resources, potentially enlisting a significant portion of the entire country's computing power for this action). In some cases, the true victim of the attack might not be the actual target of the attack packets, but others who rely on the target's correct operation. For example, in September 2002 there was an onset of attacks that overloaded the Internet infrastructure rather than targeting specific victims [6].

It also frequently happens that a DDoS attack is perpetrated accidentally, as a byproduct of another malicious activity, such as worm spread [7]. Inefficient worm-spreading strategies create massive traffic that congests the Internet and creates a denial-of-service effect to numerous clients.

While ordinary home users are less likely to become victims of DDoS attacks than large corporate networks, no one is free from the DDoS threat. The next attack may target AOL servers, denying service to many home users, or the next worm may congest the Internet so severely that no one can receive service. DDoS is an Internet-wide problem and all parties should cooperate to find a suitable solution.

2.2 DDOS ATTACKS TOOLS

Distributed denial-of-service attacks exploit different strategies to exhaust the resources of the victim. Some protocol implementations have bugs that allow a few malformed packets to severely degrade server or network performance. The victim can frequently prevent these attacks by implementing various patches to fix the vulnerability, or by filtering malformed packets. On the other hand, DDoS attacks can leverage the power of many distributed machines to make a massive number of legitimate requests for a service from a single host. Since these requests are legitimate, the victim cannot refuse to service them, nor can it recognize them as part of the attack until its resources are exhausted.

Attackers follow trends in the network security field and adjust their attacks to defeat current defense mechanisms. Spoofing of source addresses is used to avoid traceback and decoy packets and encryption are used to combat signature-based detection. The following is a quick overview of the several well-known DDoS attack tools in order to illustrate the variety of mechanisms deployed.

Trinoo [8] is a simple tool used to launch coordinated UDP flood attacks against one or many IP addresses. The attack uses constant-size UDP packets to target random ports on the victim machine. The master uses UDP or TCP to communicate with the slaves. This channel can be encrypted and password protected as well. Trinoo does not spoof source addresses although it can easily be extended to include this capability.

Tribe Flood Network (TFN) [9] can generate UDP and ICMP echo request floods, TCP SYN floods and ICMP directed broadcast (e.g. Smurf). It can spoof source IP addresses and also randomize the target ports. Communication between masters and agents occurs exclusively through ICMP_ECHO_REPLY packets.

Stacheldraht [10] combines features of Trinoo (master/agent architecture) with those of the original TFN (ICMP/TCP/UDP flood and "Smurf" style attacks). It adds encryption to the communication channels between the attacker and Stacheldraht masters. Communication is performed through TCP and ICMP packets. It allows automated update of the agents using rcp and a stolen account at some site as a cache. New program versions will have more features and different signatures to avoid detection.

TFN2K [11] is the variant of TFN that includes features designed specifically to make TFN2K traffic difficult to recognize and filter. Targets are attacked via UDP, TCP SYN, ICMP_ECHO flood or Smurf attack, and the attack type can be varied during the attack. Commands are sent from the master to the agent via TCP, UDP, ICMP, or all three at random. The command packets may be interspersed with any number of decoy packets sent to random IP addresses to avoid detection. In networks that employ ingress filtering as, TFN2K can forge packets that appear to come from neighboring machines. All communication between masters and agents is encrypted and base-64 encoded.

The **mstream** [12] tool uses spoofed TCP packets with the ACK flag set to attack the target. Communication is not encrypted and is performed through TCP and UDP packets. Access to the master is password protected. This program has a feature not found in other DDoS tools. It informs all connected users of access, successful or not, to the handler(s) by competing parties.

Shaft [13] uses TCP, ICMP or UDP flood to perform the attack and it can deploy all three styles simultaneously. UDP is used for communication between masters and agents and messages are not encrypted. Shaft randomizes the source IP address and the source port in packets. The size of packets remains fixed during the attack. A new feature is the ability to switch the master's IP address and port during the attack.

The **Code Red** [14] worm is self-propagating malicious code that exploits a known vulnerability in Microsoft IIS servers for propagation. It achieves synchronized attack by preprogramming the onset and abort time of the attack, attack method and target addresses (i.e. no master/agent architecture is involved). In addition to sophisticated attack tools, attackers also use a set of automated scripts for self-propagation of malicious code and different techniques to mask the code at the source machine.

2.3 DDOS DEFENSE TOOLS

Much of the work related to DDoS defense has been carried on in the area of intrusion detection. Intrusion detection systems detect DDoS attacks either by using the database of known signatures or by recognizing anomalies in system behavior. One might argue that it suffices to take a well-developed network-based intrusion detection system and deploy it on the source side, reversing its functions to examine outgoing traffic. However, the attack appears different at the source and at the target network. At the target network all DDoS flows converge and affect the system greatly so that detection is inevitable; at the source end those flows are still dispersed and can appear as a set of perfectly valid transactions. It is usually the sheer amount of these transactions that saturates the target network.

Several popular network monitors perform mostly signature-based detection with some limited statistical processing such as CISCO's NetRanger, Network Intrusion Detector, SecureNet PRO [15], RealSecure [16], etc. Most of these systems do not take automated action to stop the attack, but just raise an alert to the system administrator.

In [17] an on-off control approach is proposed to fight DDoS attacks in a target network. In control theoretical approach, a router detects that it is the target of a DDoS attack by monitoring its buffer queue size. Once the queue size grows over a specified threshold, the router reacts by switching to a different mode of operation and throttling incoming traffic. Only packets belonging to certain type of traffic identified as problematic are dropped. When the queue size is reduced, throttling is stopped. This approach is similar to the RED congestion control mechanism [18]. Its application to the intrusion detection domain could efficiently protect the target of the attack from overflowing its buffers by early detection of DDoS attack, while at the same time not completely cutting off the traffic from the source network. However, its application at the source network of a DDoS attack would not be so effective, since the originating network does not experience sufficiently high traffic loads to trigger the response. It should also be noted that high traffic at a source network cannot be regarded as reliable sign of DDoS attack – it could be an excess amount of legal traffic that the destination network can easily handle.

Few researches [19] have recognized that the analysis of TCP/IP packet streams through the network could be beneficial to fighting intrusion attacks. It combines a signature based approach to IDS with statistical analysis for anomaly detection. EMERALD has many similarities with our system in that it performs statistical analysis of network traffic and is able to take an automated response when the system intrusions are detected. However, its statistical subsystem would have to be substantially modified to detect anomalies in two-way traffic.

CISCO routers have built-in features such as debug logging and IP accounting that can be used for characterizing and tracing common attacks [20]. An access list can be configured to log many network events, e.g. to count packets received and categorize them by type, and to log sources of packets that are of special interest. This feature only gathers statistical data and offers no automated analysis or response.

Several researches have been proposed to augment routers with the ability to detect and control flows that create congestion and that are frequently part of DDoS attack. Flows are detected by monitoring the dropped packets in the router queue and identifying high-bandwidth aggregates that are responsible for the majority of drops. The rate limit is then imposed on the aggregate. If the congested router cannot control the aggregate itself, it can ask the adjacent upstream router to impose the rate limit on that aggregate. This upstream rate-limiting is called pushback and can be recursively propagated until it reaches the routers in the source networks [21]. This approach offers a powerful tool to not only combat DDoS attacks but to also locate and remove network congestion caused by any other reason. However, it requires significant augmentation of the routers on the whole path from the victim to the sources. A single legacy router on the path complicates the scheme and imposes the need for secure communication of non-adjacent routers which makes the system vulnerable to attacks. The approach also requires cooperation of different administrative domains, which is currently hard to achieve.

In addition, there was a heuristic and a data-structure (MULTOPS) that network devices can use to detect DDoS attacks [22]. Each network device maintains a multi-level tree that monitors certain traffic characteristics and stores data in nodes corresponding to subnet prefixes at different aggregation levels. The tree expands and contracts within a fixed memory budget. The attack is detected by abnormal values of packet ratio, and offending flows are rate-limited. The system is designed so that it can operate as either a source-end or victim-end DDoS defense system.

Moreover, several systems combat DDoS attacks by using traceback mechanisms to locate attacking nodes. These systems provide the information about the identity of attacking machines, but do not themselves stop DDoS attacks. The complexity of traceback mechanisms is large if the attack is distributed, and they may be susceptible to attempts by attackers to deceive them. Nonetheless, a good traceback system would complement the system very effectively.

Several filtering mechanisms have been proposed to prevent spoofing of source address in IP packets [23]. While IP spoofing is not necessary for DDoS attacks, it helps the attackers to hide the identity of attacking machines so they could reuse them for future attacks. Eliminating the IP spoofing would complement nicely the proposed system. It would facilitate easy distinction of normal from attack flows, and reduce greatly the damage to normal flows during attack and recovery phase.

CHAPTER 3
METHODOLOGY

3.1 PROCEDURE IDENTIFICATION

A system methodology is defined as a collection of procedures, techniques, tools and documentations aids which will help developers in their effort to implement new system. A methodology will consist of phases, themselves could also consisting of sub-phases, which will guide the developers in their choice of techniques that might be appropriate at each stage of the project and also help them plan, manage, control and evaluate or validate the system.

Thus, each project will need a methodology as it shows every step taken undergoing overall phases of research and project work. For this reason, the author decided to apply a methodology which consists of several phases to accomplish the project right on time. This model is built and revised to suit the framework of Software Development Life Cycle (SDLC) and the project itself. This model is the combination of sequential and iterative approaches of the software development process, which comprises of six main phases, which are initial planning phase, requirement definition phase, system design phase, coding and testing phase, implementation phase, and lastly maintenance and support phase. This will be clearly describes in the Figure 3.1. The details for each phase will be discussed in the next few sections.

Referring to the methodology that has been chosen, the author had created the project timeline to ensure that all tasks complete systematically within the timeframe.

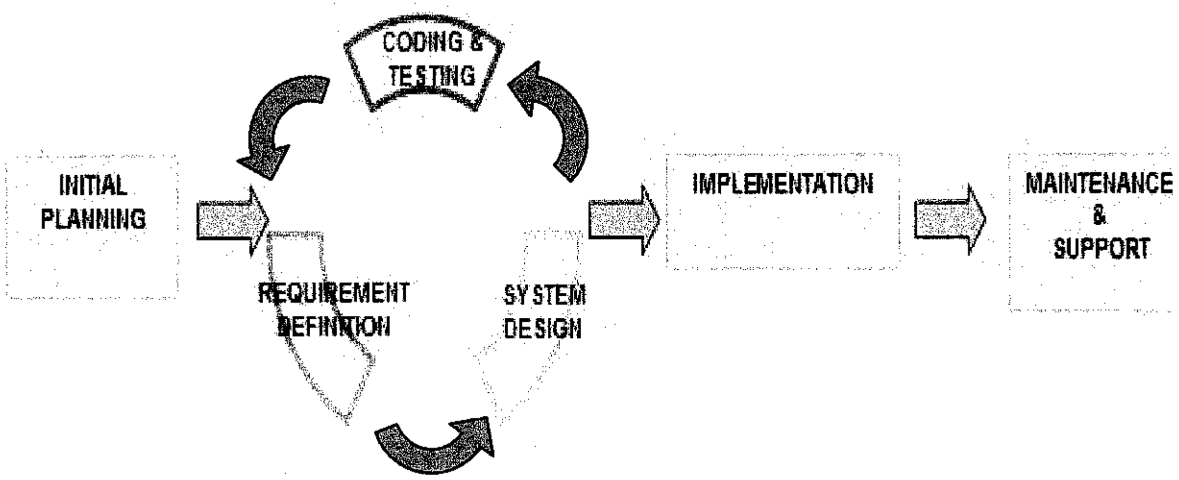


Figure 3.1: Project Framework.

3.1.1 Initial Planning Phase

This phase basically comprises with several sub-phases which are; (1) Project Proposal Submission. (2) Project Scheduling and (3) Preliminary Research. This phase is very basic phase that the author needed to accomplish throughout the first semester of final year, whereby the author has produced the prototype of the system at the end of that semester. The details of each sub-phase are described in the following section.

3.1.1.1 Project Proposal Submission

At the first-half semester of this project duration, the author had submitted proposal for this project for the purpose of getting the approval from the FYP committee, thus allowing the author to use the facilities of the University, especially the computer network facilities available in the computer labs in completing the project.

3.1.1.2 Project Scheduling

The author has scheduled the project timeline to ensure that the project will be completed just right in time. However, along the time duration of project completion, the schedule needed to change for several time for certain circumstances, whereby these included inadequate information and resources for project completion process, changes in date of report submission and project presentation, and others.

3.1.1.3 Preliminary Research

The preliminary research basically involved with the data gathering and data analysis that relevant with this project. This is also the phase whereby the author does all necessary research on the previous work of other researchers from various techniques and aspects, especially in the field of network security. To ensure that the information is adequate, relevant and not obsolete to this current of time, the author gather all the important papers from the network security proceedings held on few years back. Further, the author has analyzed and segregated collected information as they will aid in system design and development phase.

3.1.2 Requirement Definition Phase

The purpose of conducting this phase is basically to identify the problems, opportunities and directives that triggered the project, thus access the risk of pursuing the project. This phase consists of two major tasks which are defining project scope and defining project constraints. These will lead the author to list, thus analyze the essential requirements for this project.

3.1.2.1 Defining Project Scope

All the project scope should be clearly defined in order to know from where to start and finish the project development. It is important because the development or project progress should not go out the scope boundary, which means it may not less or beyond from the user or system expectations and requirements for the whole project.

For this project, the primary scopes are defined as the project will be implementing a source router approach and being implemented in Linux router or firewall. Thus, whenever the system is attempted to be implemented in other platform, it would requires huge modification, or otherwise the system will performing unexpected or failure function.

3.1.2.2 Defining Project Constraints

The author needed to clearly identify the project constraints that may disturb some features of the system from operating at optimum level. Thus, it will give the significant indication to the examiners or evaluators, or even the testers of the limitation of the project, and with that the testing or evaluation will be not beyond that limitation.

As the name applied, this project will only try to stop the attack from the source network. Or in the other word is that preventing the nodes in the source network from engaging with the cooperation to launch the DDoS attacks to the victim site.

3.1.3 System Design Phase

The purpose of the design phase is to transform the gathered information from the early phases into design specification for construction. In other word, the system design phase addresses how technology will be applied in the new system developed. Design requires soliciting ideas and opinions from the users, vendors and IT experts.

Moreover, design also requires adherence to internal technical design standards that ensure completeness, usability, reliability, performance and quality. On the other hand, design phase is concerned with the technology-based views of the system's data, processes and interfaces. Design specifications can take many forms including written documentation or working computer-generated prototypes of the new system.

For this phase, the major tasks that the author needed to take into consideration were included designing the architecture of back-end processing for the system and also designing interfaces for the front-end that will ease end-users and at the same time cater functional and non-functional specification of the system.

3.1.4 Coding and Testing Phase

Three main tasks have been developed under this phase are:

i. Coding construction – building the system with designed source code.
ii. System testing – test every code and function implemented on the modules and sub-modules.
iii. System debugging – debug any error that can be found during the testing period.

The main purpose for those activities is to build and test the system in order to ensure that every requirement and design specification have been fulfilled very well. The most important this is conducting tests of both individual system components as well as the overall system in order to ensure nothing goes wrong before it can be ready for installation and implementation.

3.1.4.1 Coding Construction

This task is very crucial for the whole system development stages. It would be referring and integrating all the findings from the early phases in order to assimilate the source codes for each feature and function for the system as well as to accommodate with all project requirements on it.

This task involved the writing source codes in C for the back-end processing that need to be integrated with the Linux libraries, as well as the JAVA for the interfaces or front-end of the system. This phase also included configuration in customizing the Linux platform to be served as gateway to the specific source network.

3.1.4.2 System Testing

During this phase, several testing have been done on the DDoS Defense system in order to ensure that everything is doing fine and has been fulfilled with all the requirements. This task is considered very important and crucial upon to the system completion. Two major testing has been conducted which are:

i. Testing during Linux router implementation
   - For each configuration changed and new configuration added to the Linux while setting it as the gateway, the testing will be done to ensure that it performs at least the basic function as the gateway, means that allow connection for two different networks.

ii. Testing the components of the system and as the whole
   - This involved the testing on integration of front-end and back-end processing. The interfaces developed should be able to process the particular function whenever it has been called.

3.1.4.3 System Debugging

For this particular task, it is all about to fix any errors that can be found after the testing has been delivered. Basically, this task really forces programmers to work until they need to push themselves over the limit. Once an error is identified and detected, the debugging process should be carried out in order to fix the error.

3.1.5 Implementation Phase

This phase included with the implementation of the end-product or integrated system at the real platform to prepare the system before the real usage of end-user. Before the system can be delivered to the end-user, the author developed written documentation as well as the user manuals to aid the system users. Eventually, the system delivery is take place, whereby in this phase the author also collecting feedback from the end-users for the evaluation of the system performance, which later can be used as the guideline for future enhancement of the system.

3.1.6 Maintenance and Support Phase

This is the final stage of the system development, which is accomplished after the system has been delivered to the end-users. This will includes the task of configuring the firewall settings for Linux router customization for the event that the system need to be implemented in each source network or formatting the hard-disk of Linux PCs whereby the firewall configuration is needed before the fully installation of the system can take place.

3.2 TOOLS AND EQUIPMENTS

For this project, the following tools will be used in order to develop the project after serious comparison has been made about the advantages and disadvantages on any possible options that may be available at the current time.

3.2.1 Hardware

i. PC with Linux operating system, with 2.4 kernel version (served as the gateway), which is having 2 NICs.
ii. PC with Windows XP operating system with standard specification (served as the client, attacker and victim).

3.2.2 Software

i. Text editor for editing the source codes
ii. GCC version 3.2.2 for the C compiler
iii. Java SDK for JAVA compiler

CHAPTER 4
RESULT AND DISCUSSION

This particular project is to come out with defense system against DDoS attacks. For this matter, the implementation technique has been identified, which is commonly deployed for DDoS defense system. As for findings; this section will be discussed on implementation suitable for this project, which is a source router approach that is implemented in the Linux router. Consequently, the discussion section will be discussed on the overall designed architecture for this project.

4.1 FINDINGS

As for project, the author has found that the source router approach is suitable for implementation of this defense system against DDoS attacks. This is based on the several advantages deploying this kind of method. Further, the simpler and cheaper configuration of the setting up the Linux router are the key factors that the author identified to be very ideal that will be used in overall implementation of the project. In next sub-sections will be discussed for on this matter.

4.1.1 Source Router Approach

The Source Router Approach uses the router (hereafter called the ***source router)***, which serves as a gateway between the source network containing some of the attack nodes and the rest of the Internet, to detect and limit DDoS streams long before they reach the target. There are a few assumptions that take into consideration while developing this defense system, which are the source network is an edge network, i.e.

most of the traffic passing through the edge routers either originates within this network or is destined for some machines in this network.

Further, the source router is assumed to be able to identify the interfaces on which it receives the source network's incoming and outgoing traffic, either through some protocol or through manual configuration. In addition, all machines on the source network use this router as the "exit router" to reach a particular set of destinations. The source router is thus in a unique position to protect these destinations from denial-of-service attacks originating at the source network by arbitrating communication with them.

The source router can detect the attack by observing two-way communication between machines on the source network and the outside hosts. The source router monitors the behavior of each destination with which the source network communicates, looking for the difficulties in communication, such as reduction of number of response packets or longer inter-arrival times. These difficulties can be a sign of a denial-of-service attack against the specific destination or of severe network congestion on the route to the destination. These difficulties are detected by periodically comparing the parameter values of the two-way traffic for each destination against a predefined model of normal traffic. If the comparison reveals the possibility of DDoS attack, the source router responds by imposing a rate limit on all outgoing traffic flows for this destination. The outcome of subsequent observation intervals either confirms or refutes this hypothesis. Confirmation further restricts the allowed rate limit, whereas refutation leads to a slow increase of the allowed outgoing traffic rate, according to the degree of well-behavior of outgoing flows.

The source router detects the attack and responds to it autonomously, without communication with other routers or human intervention. More accurate attack detection might be achieved if the source router relied on a signal from the victim instead of its own incomplete observations. However, this approach requires reliable, secure and authenticated communication between the victim and the source router, which cannot be guaranteed. Also, a more complete observation and more appropriate responses might be possible if all border routers in the source network were allowed to exchange information.

4.1.2 Linux Router Implementation

Routers are amongst the most crucial components of the Internet, as each bit of information on the Internet passes through many routers. Most of the routers used on the Internet are made by Cisco. Although these have good performance, they come at a high price. However, the performance of the Linux router makes it an attractive alternative when concerned with economizing.

The Linux router is easy to handle and configure. It does not require any special care for its use other than that required for a normal PC. If there is a problem, configuring it only takes a few minutes. Moreover, it is basically software on a floppy disk; if the Linux Router Project (LRP) box gets damaged because of power fluctuations, it can be simply repaired by instantly convert another available PC into the router by adding NICs from the corrupted LRP (if they are not corrupted) and boot it off the floppy disk. No configuration will be required for this router at all, except the runtime configuration. These are the reasons for author on deciding to implement the defense system on the Linux router, which is then act as the gateway to the designed network for the project.

There are a large variety of hardware and software products available in the open market as it has the complete structure of the ordinary Linux operating system. Additionally, having Linux as the operating system on router gives the author the extra advantage that the author can build packages according to specific needs using shell scripting.

In addition, the implementation will consist of two parts:

i The user-level implementation of the monitoring and rate-limiting components.
ii The loadable kernel module implementation of a traffic-policing component.

4.2 DISCUSSION

The author has designed the architecture of the defense system against DDoS attacks in such a way that will help the author in completing all of the stages of the system development. Though the architecture looks simpler, however it took lengthy procedures to complete the system. As a matter of fact, the system still on its ongoing process, thus the complete testing cannot be performed yet. The next sub-section will then provided with detail explanation on the system's architecture.

4.2.1 System's Architecture

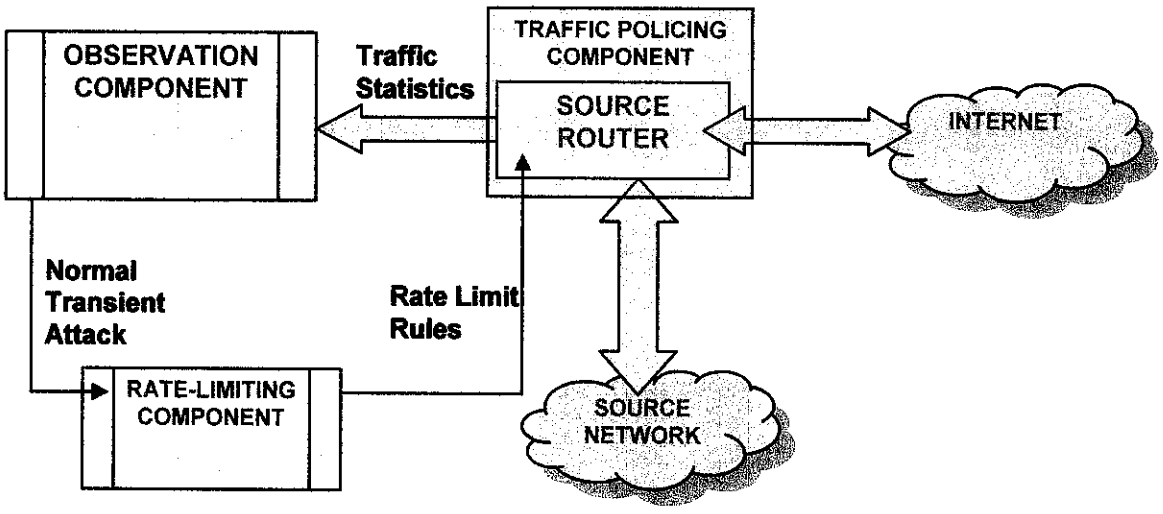


Figure 5.1: Architecture of DDoS Defense System

As shown on the figure above, the system will be having of *observation, rate- limiting* and *traffic-policing* components. The traffic-policing component must be part of the source router, while the observation and rate-limiting components can be part of a self-contained unit that interacts with the source router to obtain traffic statistics and install rate-limiting rules.

The observation component monitors all packets passing through the source router and gathers statistics on two-way communications between the police address set and the rest of the Internet, recording the flow and connection granularity. This monitoring can be performed, for example, by sniffing the traffic at the source router interfaces. Periodically, statistics are compared to models of legitimate traffic, and flows and connections are classified. Classification results are passed to the rate-limiting component, which adjusts the rate limit rules.

The traffic-policing component periodically receives rate-limited flow information from the rate-limiting component and connection classification information from the observation component. It uses this information to reach a decision whether to forward or drop each outgoing packet.

4.2.2 Philosophy

Due to the similarity of attack to legitimate traffic, it is unwise to base any defensive action on per-packet observations. Therefore, the architecture of the system is designed in such a way to have an observation of flow and connection behavior and classifies the complete flow or connection as legitimate or attack. It adjusts its operation dynamically to match the classifications.

The system detects outgoing DDoS attacks by monitoring two-way traffic between the source network and the rest of the Internet. The system looks for the following anomalies in traffic dynamics that may be signs of a DDoS attack:

- ***Non-responsive foreign host: Aggressive sending rate coupled with low response rate.*** This anomaly pertains only to two-way communications that follow a request/response paradigm such as TCP, some types of ICMP traffic, DNS traffic, etc. In these communications, one party sends one or several packets to the other party, and waits for a reply (either confirmation of receipt or a response) before sending any more packets. For such communications it is anomalous to observe an aggressive sending rate coupled with a low response rate. A low response rate is perceived by the system as an indication that the foreign host may be overwhelmed by the attack and cannot reply, while an aggressive sending rate indicates that local hosts are likely participants in the attack. By detecting non-responsive foreign hosts, the system actually aims to detect the occurrence of the denial-of-service effect.

- ***Presence of IP spoofing.*** The system deploys ingress filtering and discards, at all times, outgoing packets that do not carry local addresses. In addition, the system monitors the number of simultaneous connections between the source network and each foreign host. Those foreign hosts that are engaged in a suspiciously high number of connections with the victim are declared to be part of a subnet spoofing attack (where the agent machine spoofs addresses local to its own subnetwork).

Once the attack has been detected, the system responds by rate-limiting the total outgoing flow from the source network to the victim, and thus relieves the victim of a heavy traffic volume. As a liberal response, rate-limiting is chosen instead of filtering. Since the attack detection may be inaccurate, rate-limiting allows some packets to reach the victim and possibly correct the future detection. Instead of deploying a fixed rate limit, the system attempts to determine (guess) the maximum sending rate that the foreign host can handle.

While imposing a rate limit on total outgoing flow to the victim, this system attempts to detect those connections within the flow that are likely to be legitimate. Packets belonging to legitimate connections will be forwarded to their destination regardless of the rate limit, thus providing good service to legitimate traffic during the attack. Differentiation of legitimate from attack connections is a very difficult problem in DDoS defense. The system will be performing the differentiation in two ways:

1. It monitors connections at all times, and uses legitimate connection models to detect legitimate connections. Once detected, those connections are recorded in the *Legitimate Connection List*. When the attack is detected, packets belonging to connections from the *Legitimate Connection List* are deemed legitimate, and are not subject to rate-limiting.

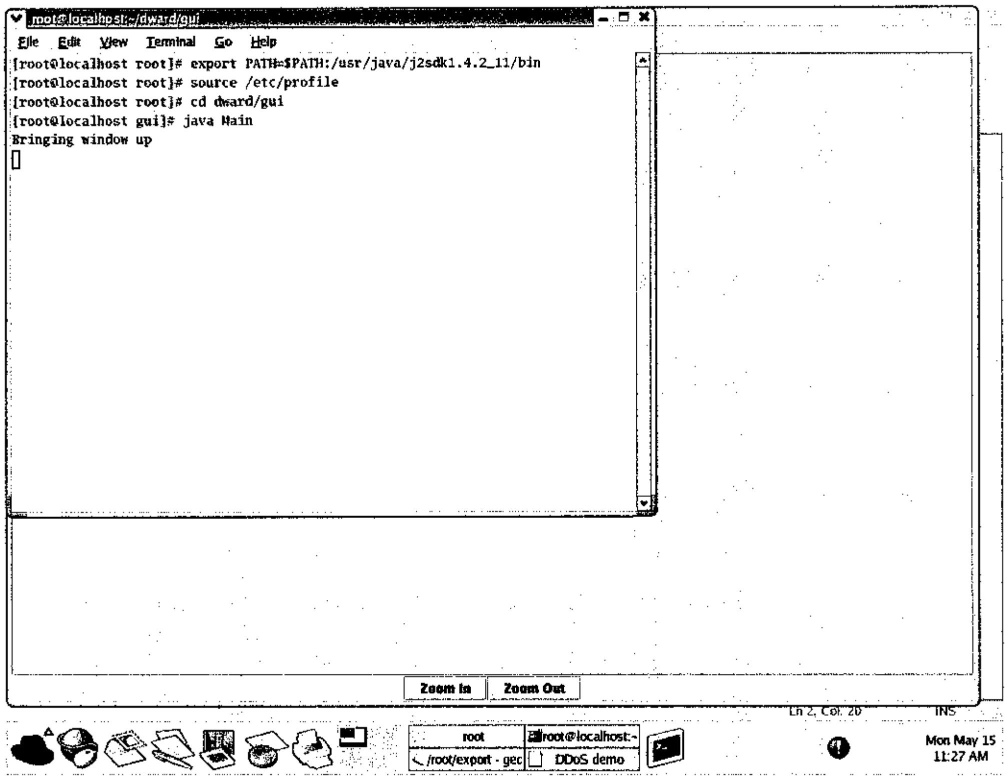
2. During the attack, the system will uses a set of models to evaluate legitimacy of packets initiating new connections. Those packets that pass the match are subject to separate rate-limiting. This technique increases the chances of successful connection origination during the attack, while limiting the amount of damage that a stealthy attacker can do (an attacker that can generate packets that pass the legitimacy test).

In order to detect both attack flows and legitimate connections, the system will uses a set of legitimate traffic models. In the flow classification case, those flows that clearly mismatch the corresponding models are deemed attack flows. In the connection classification case, those connections that clearly match the model are deemed legitimate.

4.2.3 Overview of the Developed System

This section provides with a brief overview and the screen-shots of the system interfaces, which is developed to ease in user navigation and give sense of user-friendly to the system.

As shown in the Figure 5.2 and 5.3, these two are the interfaces captured by the author when the end-user is first starting the defense application.



```
[root@localhost root]# export PATH=$PATH:/usr/java/j2sdk1.4.2_11/bin
[root@localhost root]# source /etc/profile
[root@localhost root]# cd dward/gui
[root@localhost gui]# java Main
Bringing window up
```

Figure 5.2: The user start the application

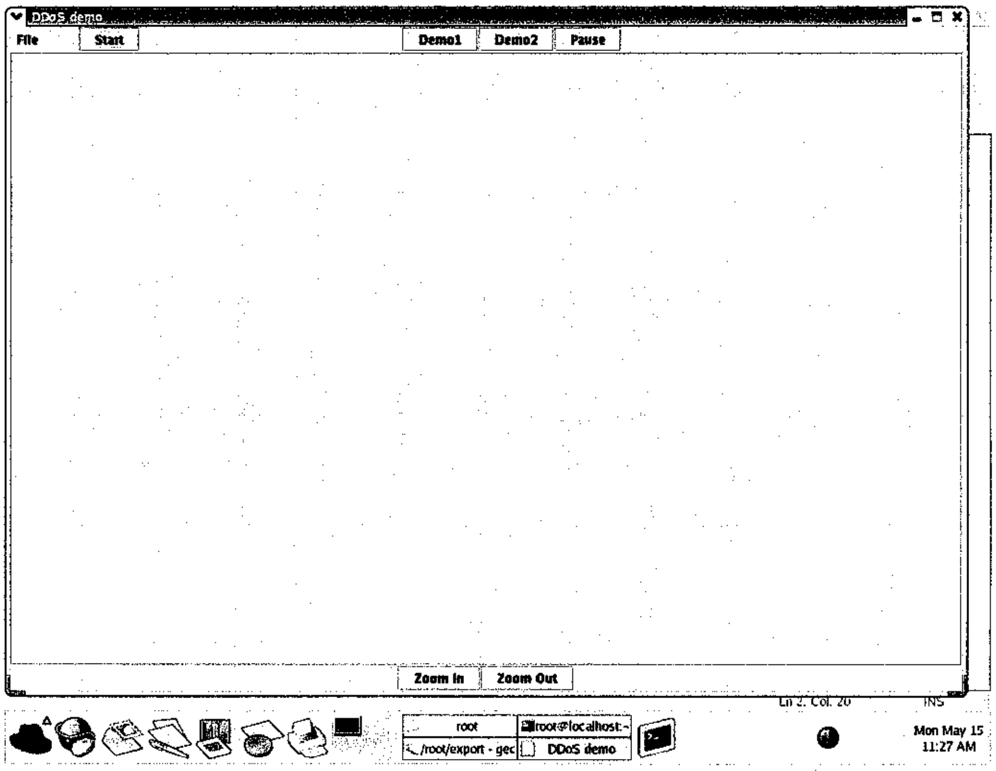


Figure 5.3: The main interface once the system is launched.

Before the testing on the application could be take place, the user needs to load predefined network topology to be tested. For this project, the simple network topology is designed that consist of five machines. Those need to be configured first in participating in the testing. The machines comprise a PC for attacker, that the attacks will be originated, a PC serves as client, that attempt to sending legitimate service through the DDoS-D PC, whereby the developed defense system been installed. Finally, a PC to be dedicated as a victim for those attacks. The overall topology described is shown in Figure 5.4.

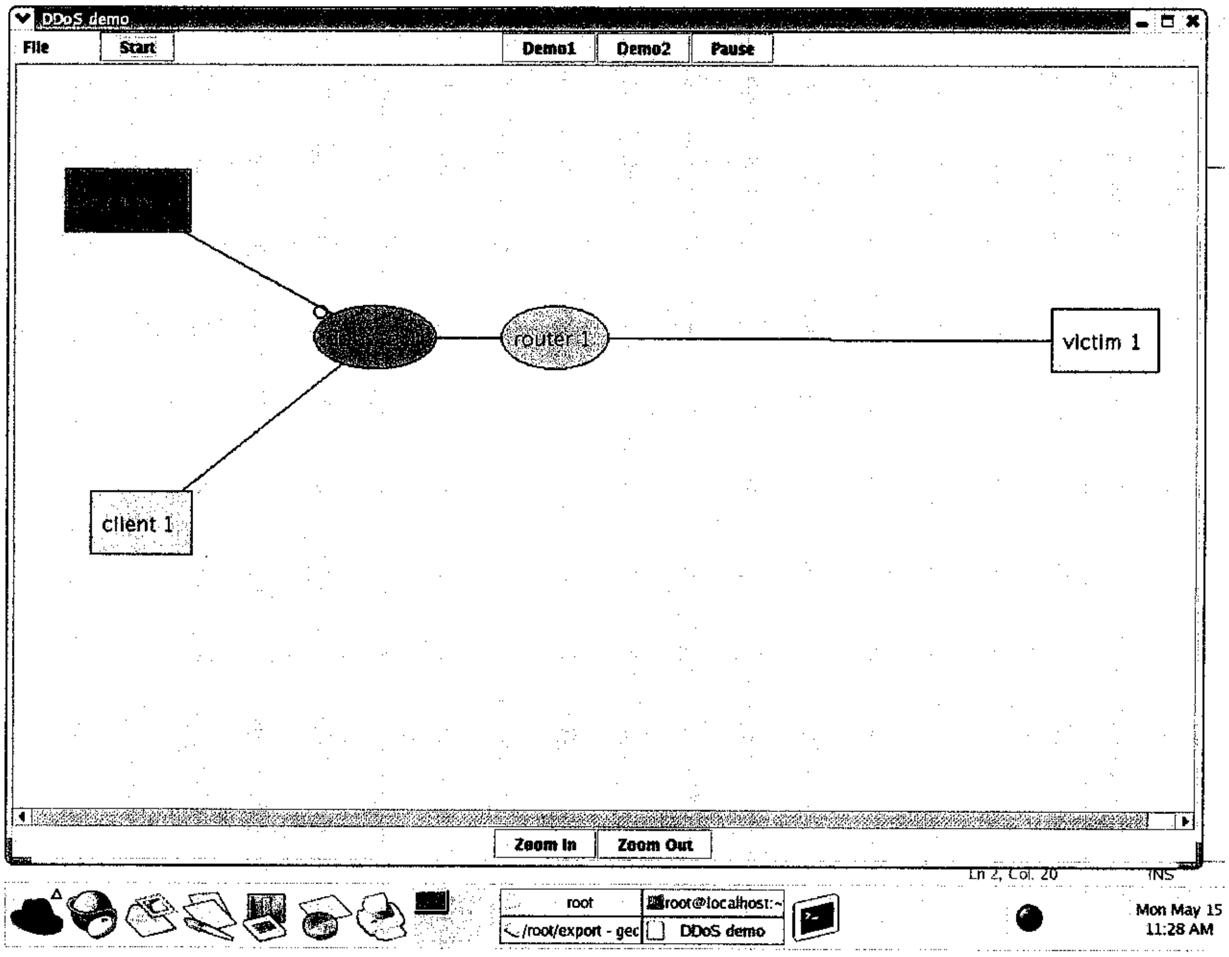


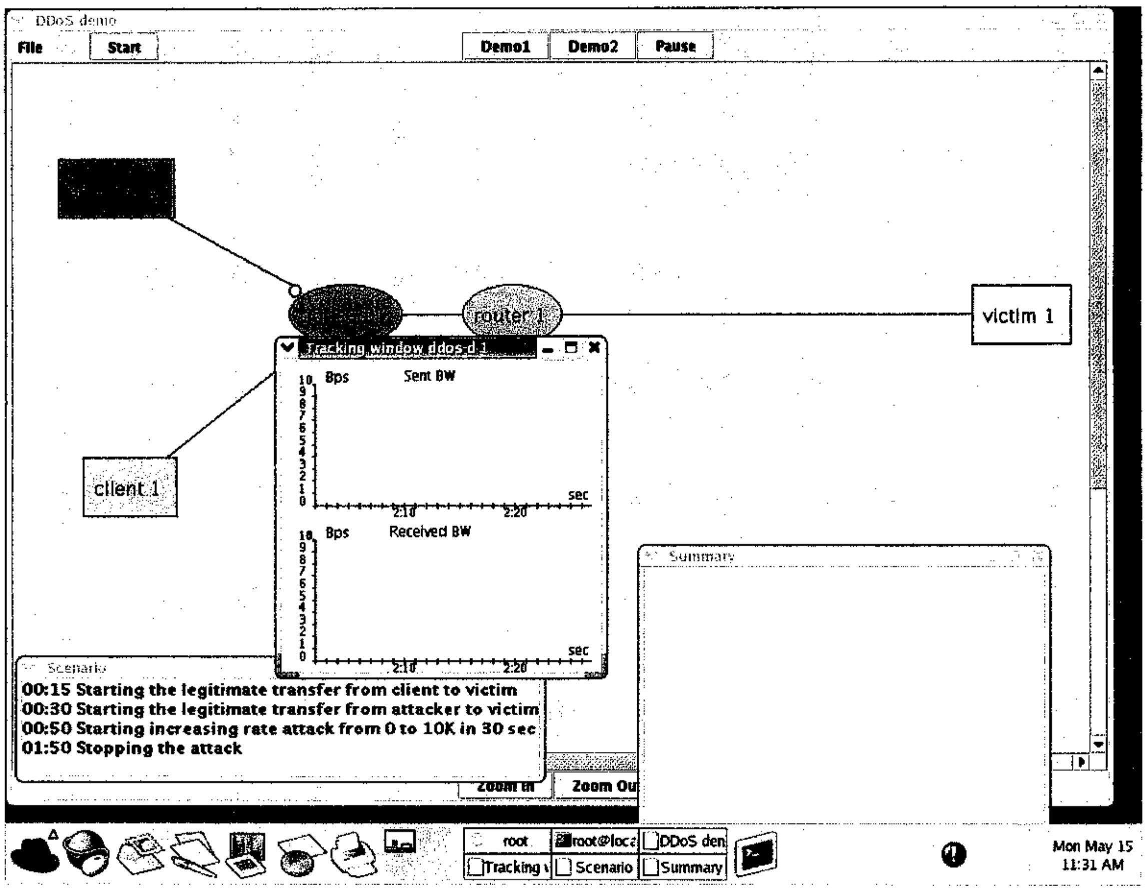
Figure 5.4: The topology of network that is being tested is loaded to the system's interface.

This simple network topology also can be referred to a single DoS attack, since only one PC is dedicated as attacker (Note that, in order to launch the DDoS attacks, multiple PCs in different networks are needed). However, in order to reach at the stage that this system would be able to stop DDoS attacks, a very essential step is to ensure that the system is able to stop a single DoS attack from one particular attacker machine must be taken first.

The system will be experiencing several modification and enhancement phases until it can be tested against complex network topology that might be comprised with multiple attacker machines in different network node.

In addition, when the testing is started, three frames appear to show the progress of the testing. There are Scenario frame, Tracking Window frame and Summary frame. The Scenario frame will be list down the activities of the testing in the timely manner. Whereas, the Tracking Window will shows the graph of traffics on that particular time of testing take place. Finally, the Summary frame should summarize the state of the flow and connection, either bad or good. These are shown in the Figure 5.5. Except, at the time the author captured the screen-shot, there was no testing process being done.

However, due to several limitations that have been identified, thus the complete testing to the developed defense system still cannot be performed. More explanation regarding this matter will be explained in the next section.



00:15 Starting the legitimate transfer from client to victim
00:30 Starting the legitimate transfer from attacker to victim
00:50 Starting increasing rate attack from 0 to 10K in 30 sec
01:50 Stopping the attack

Figure 5.5: The interface of scenario frame, tracking window and summary frame.

4.2.4 Limitation of the Developed System

At this current stage, the system is unable to perform packet capturing function, thus the system is unable to perform comparison between the good and bad traffic (attack vs. legitimate). The function of capturing the incoming and outgoing traffic can be best implemented if the whole solution working inside of the kernel.

One approach to this is Netfilter tool. Netfilter is a tool for packet interception and influencing (so called "packet mangling") outside the normal Berkley socket interface. In programming, hooking is a technique that uses a chain of procedures that act as hooks to handle an event. When an event occurs, each hook passes the execution to the next procedure in the chain until it reaches the default one. Netfilter is a set of hooks within the Linux kernel's network stack for capturing and manipulating network packets. The framework allows an administrator to define rules for dealing with network packets. These rules are grouped into chains, with each chain representing an ordered list of rules. Chains are grouped into tables, where each table is associated with a kind of packet processing.

Due to lack of technical knowledge and skills on Linux application development, the system developed doesn't meet its expected result. Moreover, the required approach seems to need lots more higher level programming and source codes.

Therefore, more study and research required to overcome those mentioned limitations in order to produce effective and sufficient defense system against DDoS attack.

CHAPTER 5
CONCLUSION AND RECOMMENDATION

5.1 CONCLUSION

DDoS attacks require the (unintended) collusion of hundreds or thousands of computers to attack a few victims and defense against DDoS attacks requires the (intended) cooperation of tens of thousands of ISPs and even consumer networks. The difficulty in handling DDoS attacks lies exactly in their simplicity. Because they misuse legitimate protocols and it is extremely difficult to separate attack traffic from legitimate traffic. By applying that defense system at the point where DDoS traffic enters the network, perhaps it helps to remove the useless load of DDoS traffic from the Internet as a whole.

As the primary objective of developing a DDoS defense system in source network, is to provide good service to a victim's legitimate clients during the attack, thus canceling the denial-of-service effect. Therefore, the DDoS defense system designed is having three components that communicate to each other to achieve the desired result. The components namely are observation, rate-limiting and traffic policing.

The system monitors the behavior of each peer with whom the source network communicates, looking for signs of communication difficulties, such as a reduction in the number of response packets or longer inter-arrival times. The system periodically compares the observed values of the two-way traffic statistics for each peer against a predefined model of normal traffic. If the comparison reveals the possibility of a DDoS attack, the system responds by imposing a rate limit on the suspicious outgoing flow for this peer.

The rate limiting and observation component will always communicating with the traffic policing component to ensure that the legitimate traffic can be forwarded unharmed in the presence of attack. However, there should have more study on the architecture of the system to ensure that it can be successfully deploy at the real implementation.

5.2 RECOMMENDATION

Several additional and more advanced functionalities can be added as future enhancement to the expansion and continuation of the defense system developed:

1) Since the system is deployed a source-end based of defense system, which is strongly detect, limit and thus stop the ongoing attacks from the source network thus prevent the victim site from been flooded with those attach. It seems like the system is strongly beneficial to the victim site and less attractive to be deployed at the source network site. However, this will be advantageous to the both networks as it can be enhance to have functionalities that would observe, detect and stop the attacks on both outgoing and incoming packets.

2) The source router approach deployed in this system would be more reliable and secured if all border routers at the source network are allowed to communicate and exchange information relating with the updating on the new attack trends and legitimate connection list, and so forth. Thus, the highly secured defense system against DDoS can be achieved.

3) More advanced feature is to allow the system to be integrated with other defense systems that might be performing in different way, in achieving cooperative with those systems. Thus, the network would be highly secured.

4) Since the system required a function of packet handling and packet capturing, therefore, it would be best if the system is embedded inside the kernel module of the Linux itself. Thus, more in-dept studies especially in technical knowledge on Linux application development required to enhance the system.

REFERENCES

[1] Kessler, G. *Defenses Against Distributed Denial of Service Attacks*. Retrieved Oct. 10, 2005, from the World Wide Web: http://www.garykessler.net/library/ddos.htm

[2] Mirkovic, J. (2003). *D-WARD: Source-End Defense Against Distributed Denial-of-Service Attacks.* Unpublished doctoral dissertation. Los Angeles: University of California.

[3] Mirkovic, J., Reiher, P., Prier, G. (November 2002). *Proceeding of ICNP 2002: A Source Router Approach to DDOS Defense.* Paris, France.

[4] Mirkovic, J., Reiher, P., Prier, G. *Attacking DDOS at the Source*. Retrieved Oct. 13, 2005 from the World Wide Web: http://www.lasr.cs.ucla.edu/ddos/

[5] S. Shankland. "SCO Web site slammed by Net attack." *ZDNet.com*, Retrieved Nov 2005 from the World Wide Web: http://zdnet.com.com/2100-1105_2-999514.html.

[6] R. Naraine. *Massive DDoS Attack Hit DNS Root Servers*, Retrieved October 2005 from the World Wide Web: http://www.esecurityplanet.com/trends/article/0,,10751 1486981,00.html

[7] D. Moore. *The spread of the code red worm (crv2)*. Retrieved October 2005 from the World Wide Web: http://www.caida.org/analysis/security/codered/coderedv2_analysis.xml.

[8] D. Dittrich. *The DoS Project's 'trinoo' distributed denial of service attack tool.* Retrieved Nov 2005 from the World Wide Web: http://staff.washington.edu/dittrich/misc/trinoo.analysis

[9] D. Dittrich. *The 'Tribe Flood Network' distributed denial of service attack tool.* Retrieved Nov 2005 from the World Wide Web: http://staff.washington.edu/dittrich/misc/tfn.analysis.txt

[10] D. Dittrich. *The 'Stacheldraht' distributed denial of service attack tool.* Retrieved Nov 2005 from the World Wide Web: http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.txt

[11] CERT Coordination Center. *CERT Advisory CA-1999-17 Denial-Of-Service Tools.* Retrieved Nov 2005 from the World Wide Web: http://www.cert.org/advisories/CA-1999-17.html

[12] D. Dittrich. *The 'mstream' distributed denial of service attack tool.* Retrieved Nov 2005 from the World Wide Web: http://staff.washington.edu/dittrich/misc/mstream.analysis.txt

[13] S.Dietrich, N. Long and D. Dittrich. *An Analysis of the "Shaft" distributed denial of service tool.* Retrieved Nov 2005 from the World Wide Web: http://www.adelphi.edu/~spock/shaft_analysis.txt

[14] CERT Coordination Center. *CERT Advisory CA-2001-19 'Code Red' Worm Exploiting Buffer Overflow In IIS Indexing Service DLL.* Retrieved Nov 2005 from the World Wide Web: http://www.cert.org/advisories/CA-2001-19.html

[15] MimeStar.com. *SecureNet PRO Feature List.* Retrieved Dec 2005 from the World Wide Web: http://www.mimestar.com/products/

[16] Internet Security Systems. *Intrusion Detection Security Products*. Retrieved Dec 2005 from the World Wide Web: http://www.iss.net/securing_e-business/security_products/intrusion_detection/index.php

[17] S. Liu, Y. Xiong, and P. Sun. (June 2000). IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop: *On Prevention of the Denial of Service Attacks: A Control Theoretical Approach*

[18] S. Floyd and V. Jacobson. (August 1993) *Random Early Detection Gateways for Congestion Avoidance.* IEEE/ACM Transactions on Networking.

[19] P.A. Porras and A. Valdes. (March 1998) *Proceedings of the Symposium on Network and Distributed System Security : Live traffic analysis of TCP/IP gateways*

[20] Cisco. *Characterizing and Tracing Packet Floods Using Cisco Routers.* Retrieved Dec 2005 from the World Wide Web: http://www.cisco.com/warp/public/707/22.htm

[21] S.Floyd et al. *Pushback Messages for Controlling aggregates in the Network.* Retrieved August 2005 from the World Wide Web: http://search.ietf.org/internet-drafts/draft-floyd-pushback-messages

[22] T. M. Gil and M. Poleto. (August 2001) *Proceedings of 10th Usenix Security Symposium :MULTOPS - A Data Structure for Bandwidth Attack Detection*

[23] P. Ferguson and D.Senie. *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing*. RFC 2827.