國 立 交 通 大 學

資 訊 工 程 學 系

博 士 論 文

分散式阻斷攻擊防禦機制之設計

Designing Protection Mechanisms against DDoS Attacks

研 究 生：李富源

指導教授：謝續平 博士

中 華 民 國 九 十 四 年 五 月

Designing Protection Mechanisms against DDoS Attacks

Student: Fu-Yuan Lee

Advisor: Dr. Shiuhpyng Shieh

A Dissertation Submitted to

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Chiao-Tung University

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science and Information Engineering

May, 2005

Hsinchu, Taiwan, R.O.C.

分散式阻斷攻擊防禦機制之設計

研究生: 李富源　　　　　　　　　　　　　　指導教授: 謝續平 博士

國立交通大學資訊工程學系

摘要

隨著電腦網路技術的不斷進展，人們的日常生活和電腦網路產生了密切的關聯，同時，網路攻擊也成了一個值得注意的議題。 近年來，分散式阻斷攻擊成了諸多網路攻擊事件中最引人注目的焦點。 攻擊者藉由入侵眾多安全防護較薄弱的電腦系統，進而利用這些被入侵的系統對網路伺服器進行阻斷式攻擊。 在攻擊期間，該網路伺服器的使用者將感受到明顯的網路延遲、大量的封包遺失，或根本無法與伺服器建立網路連線，攻擊者據此可輕易達到其阻斷服務之效果。 反之，要抵抗或偵測這類型的網路攻擊事件是非常困難的，其主要原因來自於網路上大量的防護層級較低的電腦、偽造網路位址的使用、攻擊封包與合法封包間的高相似度以及分散式網路管理所造成的困難。

本篇論文所要探討的主題為分散式阻斷攻擊的防禦技術。 為了要設計合宜且可行的防禦機制，首先我們得對分散式阻斷攻擊有深入的了解。 因此，在本篇論文的第一部分，我們針對分散式阻斷攻擊的成因、分散式阻斷攻擊的型態以及常見的攻擊程式做廣泛的分析與整理。 我們指出幾個在設計防禦機制時會面對的問題，並對目前既有的分散式阻斷攻擊防禦機制的運作和設計原理做深入的介紹和討論。

本論文的第二部分著眼於分散式阻斷攻擊防禦機制之設計。 對於該類型網路攻擊的對策，我們分別從三個不同的角度切入：受害者端的防禦、追蹤攻擊者的技術以及攻擊者端的防禦。 受害者端的防禦系統主要致力於偵測偽造來源位址之網路封包。 由於分散式阻斷攻擊的攻擊流量主要由這類型的封包組成，因此，我們可以藉由辨認並阻絕偽造來源位址之網路封包來達成過濾攻擊封包的目的，進而維護網路伺服器繼續提供服務的能力。 另外，由於攻擊者可以任意偽造來源位址，受害者無法辨認攻擊封包的來源，而攻擊者也藉此來降低被發現的風險。 為了要嚇阻攻擊者持續進行攻擊，本論文的第二個研究主題即為探討追蹤攻擊者的技術。 本論文中所提出的追蹤技術可以將攻擊封包所行經的網路路徑進行編碼，編碼後的路徑可以儲存在攻擊封包之中，和攻擊封包一起到達受害者端，或

是由幾個 ICMP 封包將路徑資訊送至受害者端。 本論文研究的第三個主題為一個可以將攻擊封包限制於攻擊者端網路的防禦技術。此項研究可以阻止攻擊封包進入網際網路，進而減少因攻擊流量而造成網路壅塞之情形。

在本論文中所提出的這三個防禦機制可協助受害者過濾攻擊封包、追蹤攻擊者位址以及可將攻擊封包局限於攻擊者端網路。 根據實驗結果，本文所提之三個系統表現優於目前存在之防禦機制。 除此之外，這三個系統都可用漸進的方式佈建於現有網際網路中，亦可和現有的路由器系統共存，並僅須對路由器進行少量的修改即可支援文中所提出之防禦機制。換言之，我們所提出的防禦機制可被部署於大型的網路系統，並可被視為建置分散式阻斷攻擊防禦機制時，幾個非常重要且有效的基石。

關鍵詞: 網路安全、入侵偵測以及防禦、分散式阻斷攻擊、偽造網路位址、攻擊來源追蹤技術

Designing Protection Mechanisms against DDoS Attacks

Student: Fu-Yuan Lee Adviser: Dr. Shiuhpyng Shieh

Department of Computer Science and Information Engineering

National Chiao Tung University

ABSTRACT

The widespread incidences of distributed denial-of-service (DDoS) attacks have highlighted a great demand for effective DDoS countermeasures. Owing to a large number of insecure systems supplying DDoS attackers with abundant attack zombies and the set of easily acquired and deployed DDoS attack tools, malicious users can easily overwhelm Internet servers with DDoS attack packets. On the other hand, the defense of DDoS attacks has been made very complicated by large sets of attack zombies, IP spoofing techniques, high level of similarity between legitimate and attack packets, and the independent and distributed nature of network administration.

The work presented in this dissertation is on the defense of DDoS attacks. To better understand DDoS attacks and to design effective and appropriate DDoS defense and response, in the first part of this dissertation, we elaborate the causes of DDoS attacks, types of attack streams, and commonly used attack tools. Afterwards, we move on to identify fundamental challenges to DDoS defense and investigate existing approaches to DDoS attacks.

The second part of this dissertation concerns with the design of DDoS defense mechanisms. In particular, it explores defensive approaches from three distinct directions, namely victim-end defense, attack traceback and attacker-end defense. The proposed victim-end defense scheme aims at identifying spoofed IP packets which dominate DDoS attack traffic. This allows Internet servers to sustain their services to legitimate clients when under attack. With the presence of IP spoofing, the source IP addresses inscribed in DDoS attack packets are usually untrustworthy, and DDoS attackers run at low risk of being discovered. To deter future DDoS attacks, it is imperative to locate origins of DDoS attack flows, and therefore the second focus of our research is to trace the sources of spoofed DDoS attack flows even if there is only one single packet in each attack flow. With the proposed traceback scheme, an Internet path traversed by an IP packet can be encoded with a small number of bits, and transmitted to the packet's destination along with the packet itself or in a few additional ICMP packets. The third defense approach targets at detecting DDoS attack flows at their sources and confining attack packets at source networks. With a widespread deployment of this scheme, we can stop attack packets from entering the Internet and subsequently reduce possible network congestions caused by attack streams.

Schemes presented in this dissertation allow victim servers distinguishing spoofed attack packets from legitimate ones, support identifying attack sources and help confine attack flows at their sources. In light of our experimental results, the proposed schemes outperform existing DDoS defense approaches. Furthermore the proposed schemes can be deployed incrementally and can coexist with legacy routers. Very little enhancements to Internet routers are involved. With these characteristics, the proposed schemes can be deployed gradually on large networks, such as the Internet, and are considered useful building blocks for constructing effective defense for DDoS attacks.

Keywords: Network Security, Intrusion Detection and Prevention, Distribution Denial-of-Service (DDoS) Attacks, IP Spoofing, IP Traceback

Acknowledgement

回首這六年博士班生涯，雖顛躓走來一路跌跌撞撞，所幸路上不時有人給我扶持，為我指引方向，在我失志時給我鼓勵，在我略有所得時和我分享。 感謝這二千多個日子以來所有幫助過我的人，謝謝你們，有了你們，我才能在這過程中有所成長。 而今我即將走進博士班生涯的終點站，對於你們所有的付出，我，懷著滿心的感謝。

在邁向下一階段旅程之際，在這裡，我要特別感謝我的指導教授謝續平老師這幾年來的教誨與照顧。 從大學部專題、碩士班，到博士班這八年來的指導，使我能略窺電腦及網路安全之堂奧，若非謝老師在學術研究以及日常生活上給我諸多的幫助，這份學位論文就難以順利完成。 此外，也要萬分感謝賴溪松教授、雷欽隆教授、詹進科教授、黃世昆教授、官大智教授及邵家健教授等口試委員，在百忙中撥冗，不辭遠道而來，提供諸多寶貴的意見和指正，使本論文能略臻於完備，也讓我更清楚未來要努力的方向。

我還要感謝「分散式系統與網路安全實驗室」的所有成員，感謝他們無論是在課業的切磋或是生活經驗的分享上，都讓我渡過充實而美好的一段時光。 其中，林宸堂學長及楊文和學長常不吝與我分享他們的經驗和研究心得，讓我在博士班就讀過程中受益匪淺。 另外，也要特別感謝楊明豪博士和士一的幫忙，不管是我的研究工作或是實驗室的日常事務，少了他們的力量，很多事情都沒辦法順利地完成。 衷心地感謝他們這幾年來在各方面給我的建議與協助，讓我能用更多的時間加速完成這份論文。

最後，我要以這份博士學位的榮耀獻與我最親愛的父母與家人。 感謝我的父母對我多年來的照顧與支持，費盡苦心為我鋪設受高等教育的道路。 他們的辛勞，我永遠銘記在心，我知道我現在能擁有的一切都是用他們多年的汗水換來的，這份恩情實難以用言語文字表達。 我也要感謝我的大哥、大妹和二妹，這些年來，我常因課業而無法善盡為人子的責任，多虧有了他們，使家中一切安好，並代我承歡膝下，才讓我能無後顧之憂地攻讀博士學位。 另外，我要感謝陪著我近十年的女友怜儀，感謝她耐心地伴我渡過生命中的低潮，容忍我易焦躁不安的情緒，幫助我維持心境的平和，給我繼續往前走的勇氣。 在學業上，也由於有了怜儀的鼓勵，我才能在困惑中冷靜地找到突破的方向。 也謝謝我家的小狗 Dolly，每次回家看著牠搖尾巴、跳來跳去高興的樣子，總在不覺中減輕了我的壓力，為我的生活注入許多快樂的元素。 感謝老天的安排，給我學習的機會，讓我在再出發時有謙卑的心境和充分的信心。 回首這二千多個日子，一切得來不易，此刻的心情除了喜悅，還有更多的感恩，我願以心中的喜樂與所有曾幫助我、關心過我的人共享，也願今日這博士學位所代表的一切，能略略安慰父母與家人多年的辛勞與殷殷期盼之心。

李富源　2005 年 5 月 30 日于新竹交通大學

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Denial-of-service (DoS) attacks [18,52,80], which intend to disable target Internet servers from providing services to their clients or to degrade quality of services, have been considered a major threat to the stability of the Internet. When interacting with an Internet server under DoS attack, legitimate clients would experience abnormally long network latency or high packet loss rates. In some cases of DoS attacks, legitimate clients may even be deprived of network connectivity to the server [16], or the server may crash and then all services running on it are suspended [17,59]. As the Internet have increasingly become an important communication infrastructure for many business activities, DoS attacks can lead to not only disturbance in network communications but also huge financial costs. Commercial web sites would suffer considerable revenue losses if their web servers are forced to shut down or are overloaded with a multitude of malicious HTTP requests. Furthermore DoS attacks can also adversely interfere with some crucial communications that must be completed in a timely manner.

Owing to the ease of obtaining DoS attack programs and the user-friendly interfaces provided by these tools, launching DoS attack needs no advanced techniques in programming or system administration. An unsophisticated user can easily acquire attack programs and then employ these tools to exploit design weaknesses in communication protocols or in infrastructure components on the Internet. Due to the stateless nature of destination-based best-effort delivery service provided by the Internet, source IP addresses in attack packets are untrustworthy. Attackers run at low risk of being discovered and punished. These factors consequently result in a huge population of so-called "script kiddie" (i.e. unsophisticated users who are capable of manipulating attack scripts to conduct DoS attacks) and contribute to a widespread incidence of DoS attacks.

Roughly DoS attacks can be classified into two categories: **vulnerability-based attacks** and **flooding-based attacks**. The former consists of attacks exploiting vulnerabilities in software (including service programs, software utilities, operating systems and configurations of them) to disable chosen Internet services. The latter refers to attacks paralyzing target services by over-consuming resources that are critical for the services to function properly. Software vulnerabilities mostly originated from implementation errors or mis-configurations. By taking advantage of software vulnerabilities, an attacker usually can gain additional privileges on target systems or even crash them with small numbers of attack packets. These attack packets are usually mal-formatted and are of special type and content. This feature allows intrusion detection systems (IDSs) detecting malicious packets by looking for attack patterns in IP packets. Moreover, software vendors will generally issue patches to fix vulnerabilities immediately after vulnerabilities are discovered. From the above-mentioned perspectives, vulnerability-based attacks normally can be prevented by secure programming techniques [4, 35] and sophisticated administration skills/tools [19, 20, 74, 110], and can be effectively detected and stopped by IDSs/firewalls and system patches respectively.

Unlike vulnerability-based DoS attacks, flooding-based DoS attacks do not rely on software vulnerabilities. Instead, they adversely affect the execution of a target Internet service by simply exhausting some critical resources, such as network bandwidth, memory space, or CPU computation power. Internet services being subject to flooding-based DoS attacks mostly have a common design weakness. That is, in general they will devote some resources in prior to, or in part of, inspecting the legitimacy of received service requests. Even if the amount of resources devoted for each service request is small, attackers can still over-consume these resources by sending a target servers a large number of malicious service requests. A crucial fact is that distinguishing malicious service requests from legitimate ones is extremely difficult for target Internet servers. Malicious service requests are usually very similar, or even identical, to requests originated from legitimate clients. The high level of similarity generally will lead to collateral damages to legitimate clients, and this constitutes a major obstacle to effective DoS defense.

Distributed DoS (DDoS) attacks refer to flooding-based DoS attacks accomplished with a enormous number of attacking machines distributed over the Internet. The major difference between DoS and DDoS attacks is in their scale. In a DDoS attack, many flooding-based DoS streams originated from multiple machines are arranged to flood selected Internet servers around in the same time. In this way, the impact of a simple DoS attack can be greatly amplified. To conduct such a DDoS attack, an attacker first needs to compromise a sufficient number of computer systems. Then at some later points in time, these compromised hosts, or simply called *zombies*, are used to flood chosen Internet servers in order to jam their Internet links, overload the servers or crash them.

The design of effective DDoS countermeasures is made very complicated by several reasons. First, the Internet is naturally vulnerable to DDoS attacks. This is mainly because the Internet was not designed with security issues in mind. Internet routers simply provide destination-based best-effort packet delivery service and generally do not have the capability to determine legitimacy of IP packets. Attackers are allowed to arbitrarily construct mal-formatted IP packets or to forge IP header fields. Currently there is nothing Internet routers can do to stop malicious users from attacking other users on the Internet. Second, DDoS attack flows are usually composed of legitimate-like packets. And finally, there will always be careless system administrators or don't-care users who expose weaknesses of their systems to attackers, and these insecure computers form a fertile ground for attack zombies. All these characteristics make DDoS attacks very effective and also very hard to defend against.

This introductory chapter first describes the demand for effective DDoS countermeasures and briefly discusses challenges in designing DDoS solutions. Then we move on to present major defensive strategies and highlight three essential issues that will be discussed in this dissertation. Design principles and criteria for evaluation of DDoS solutions are presented next. Finally key contributions of our work and the roadmap of this dissertation are given in the end of this chapter.

1.1 Motivations and Problem Statements

As aforementioned, DDoS attacks have long been regarded as a serious problem, and many research efforts have been devoted to the design of countermeasures. Herein we first present a classification of DDoS defense strategies. (Technical details of approaches belonging to each class will be discussed shortly in Chapter 2.) We then elaborate the selection of our defense approaches, and explicitly define the problems to be addressed.

Most existing DDoS defense schemes roughly fall into the following three categories: *proactive defenses*, *reactive defense* and *DDoS attack traceback*. The first category is composed of approaches which aim at eliminating fundamental causes of DDoS attacks. For instance, some such approaches attempt to hide real location (i.e. the IP address) of a protected Internet server, and at the same time, retain its service to legitimate clients. Without the server's IP address, it would be very difficult for an attacker to overwhelm the server or to jam its network connection. Usually such solutions involve changes to existing communication protocols and Internet infrastructure components Softwares/firmwares installed on switches, routers and end host systems need to be updated, and therefore these solutions generally incur tremendous efforts for their deployment.

The second field of research consists of approaches responding to DDoS attacks when an attack is ongoing. Schemes in this category actually contain both techniques for detecting the presence of DDoS attacks and responding mechanisms for mitigating the impact of attacks. Techniques for attack detection generally involve analyzing network traffic statistics, monitoring system behaviors and inspecting known attack patterns in IP packets. Responding mechanisms control resource usage, discard identified attack packets or impose rate-limiting rules on attack flows. Most of these solutions are embodied in firewalls, end host operating systems and sometimes on Internet routers, and currently these reactive approaches are dominant solutions for DDoS attacks. The final research area focuses on locating the sources of DDoS attack flows. Though attack traceback provide little help to protect Internet servers from being attacked, it is considered an essential element for automatic response to DDoS attacks and the first step to further determine attackers who manipulate recognized attack zombies. In other words, attack traceback helps construct a deterrence for future DDoS attacks.

This dissertation is on the defense of DDoS attacks. In particular, according to the classification on DDoS defense systems, our work is about the following two types of defense approaches: (1) reactive schemes that enable edge routers and end host systems to defend against DDoS attacks, and (2) traceback schemes that allow constructing the Internet paths traversed by DDoS attack flows.

One key consideration in designing effective DDoS countermeasures is to impose appropriate defense mechanisms at correct positions. Specifically, defense systems need to be deployed at or before positions where resource exhaustion occurs. Otherwise, a incorrectly-placed defense system will suffer from either the ineffectiveness in blocking attack flows or difficulties in detecting attacks. For instance, consider a DDoS attack that attempts to exhaust CPU computation power of victims. The persistent high CPU utilization at victim systems can serve as a signal of the attack, and reactive defense systems installed on victims can start to discard identified attack packets once an attack signal is captured. While, without a significant surge on the volume of network traffic or a observable anomaly in traffic statistics, it is relatively harder for defense systems installed at Internet routers to sense the occurrence of such attacks, and consequently they are weak in blocking attack flows. Consider another attack example that purports to exhaust network bandwidth of a victim network. In this example, defense systems on Internet routers can detect the presence of such attack by monitoring network traffics and suppress attack flows quickly by rate-limiting rules. On the other hand, defense installed on victims provide little help because dropping identified attack packets at victims would not help alleviate network congestions on upstream routers.

Hence, in this dissertation, we choose to develop two types of reactive DDoS defense systems, which are deployed at victim-end and source-end (or, attacker-end) respectively. The major objective of victim-end defense is to identify and discard attack packets at victims, and the source-end defense system is used to confine attack flows at their sources. The two systems complement each other and can handle a majority of DDoS attacks.

Another issue addressed in this dissertation falls into the category of attack traceback. Specifically, we focus on a difficult case of the IP traceback problem, i.e. single packet IP traceback. In a highly distributed DDoS attack, each zombie may only send a few attack packets, and most of existing traceback techniques cannot identify attack sources in this situation. This dissertation attempts to solve this problem by proposing a traceback mechanism that are capable to locate attack origin(s) even if there is only one single packet in each attack flow.

In short, we explore the DDoS problem from three directions, namely *victim-end defense*, *source-end defense*, and *IP traceback*. Three DDoS defense approach are developed and objectives of each approach are summarized below.

Victim-End Defense To stop DDoS attacks from inflicting damages on victims, a defense system deployed at victims must be able to identified and then discard attack packets at high accuracy. Because DDoS attack flows mostly comprise spoofed IP packets, the goal of proposed victim-end defense systems is to identify IP packets with spoofed source IP addresses. In this way, we can discard a majority of DDoS packets at the victim and thus sustain the service quality to legitimate clients.

IP Traceback An IP traceback mechanism for tracing individual packets is proposed. Specifically single packet IP traceback refers to the process of identifying a list of IP addresses representing the routers on the Internet path traversed by an IP packet. With the proposed scheme, information for reconstructing an Internet path is encoded with a small number of bits and can be delivered to a destination host along with each IP packet in an efficient and authenticated way. Though the proposed scheme requires support from Internet routers, it can coexist with legacy routers and can be deployed incrementally.

Source-End Defense An approach for recognizing and confining DDoS attack flows at attack sources is presented. Our approach models behaviors of legitimate network flows and construct profiles for each of them. A network flow that does not comply with existing normal flow profile is consider an attack flow, and rate-limiting rules are then imposed on attack flows so as to prevent attack packets from entering the Internet. With a widespread deployment of this scheme, DDoS attack flows can be confine at their sources.

1.2 Common Design Guidelines

Before we go into details of each proposed solutions, we first discuss several design principles that are considered important to the effectiveness of DDoS defense mechanisms.

- **Lightweight Processing:** The processing load of DDoS defense systems must be lightweight. Otherwise, a defense system itself would be the target of DDoS attacks. An attacker can disable the defense by overloading the defense system. In particular, modifications or enhancements to routers should be design very carefully so as not to cause substantial processing overhead on routers.

- **Incremental Deployment:** DDoS defense systems should avoid relying on a significant retrofitting to routers or end host systems in a all-or-nothing fashion. It is unrealistic to assume that the required retrofit can be completed within a short period of time. Instead, a feasible solution needs to support incremental deployment, which makes the defense system gains its effectiveness with respect to the degree of deployment.

- **Loose Cooperation:** A DDoS defense system had better avoid a tight cooperation assumption among different ISP networks. This is because cooperation normally involves complex coordinations among ISPs and therefore incurs substantial overhead. This will make deployment of the DDoS defense mechanism difficult in large networks, such as the Internet.

- **Accuracy:** An effective DDoS countermeasure should be accurate, in terms of low false positive ratios and low false negative ratios. Low false positives mean that the defense will not lead to significant collateral damage to legitimate traffic, and low false negatives mean that only a negligible portion of attack traffic is left undetected. More importantly, accuracy must be maintained all the time even when the DDoS defense mechanism itself is under some sorts of attacks launched by attackers who possess reasonable and sufficient resources, such as a complete topological map of the Internet and the IP addresses of the Internet routers. Attackers would try all the possibilities to circumvent the defense such that (1) attack traffic can circumvent detection and filtering mechanism, or (2) the defense mechanism will be deceived into misjudging legitimate packets as malicious ones. Therefore it is important for a DDoS defense mechanism to resist sophisticated attacks and keep its accuracy under all circumstances.

1.3 Contributions and Synopsis of the Dissertation

In this dissertation, we first present in Chapter 2 an investigation on fundamental causes of DDoS attacks, types of attacks and attack programs. An extensive literature survey on DDoS defense mechanisms are also given in the chapter. Design concepts and operational overviews of existing defensive approaches are described. This investigation helps recognize major tends in the evolution of both DDoS attacks and defense. It consequently serves as a useful foundation for developing appropriate and effective DDoS countermeasures.

The main contribution of this dissertation is on the design of three distinct DDoS countermeasures. In this dissertation, a chapter is devoted to each proposed defensive approach. For each proposed scheme, we first present background, related work and then describe the motivation, key concepts and rationales, details of the proposed defensive approach, and corresponding evaluations.

In Chapter 3, we will describe a victim-end approach for identifying IP packets with spoofed source addresses. The proposed scheme allows detecting and blocking attack packets on a per packet basis, and experimental results showed that our scheme can detect a majority of spoofed attack packets and cause very little collateral damage to legitimate clients.

In Chapter 4, we will present a traceback scheme that allows constructing an Internet path traversed by a single IP packet. An space-effective route encoding technique is presented, and with the proposed encoding, path information can be transmitted to the destination of an IP packet along with the packet, or by a few additional ICMP packets. Our scheme requires only one multiplication and one addition operation on Internet routers and can be deployed incrementally. Furthermore, analysis and experimental results showed that our scheme can identify the origin(s) of IP packets at high accuracy.

In Chapter 5, a scheme for detecting and blocking attack flows at their sources will be presented. The proposed scheme is based on an existing source-end DDoS defense system, called D-WARD. Our approach attempts to complement rather than to replace D-WARD. New criteria for detecting DDoS attacks are derived from several characteristics of DDoS attack flows. Experiment results showed that the proposed scheme can fix weaknesses of D-WARD and outperforms D-WARD in terms of detection accuracy and response time. Finally, Chapter 6 concludes this dissertation by discussing directions for future work in the field of DDoS defense.

Chapter 2

Distributed Denial-of-Service Attacks and Defense Approaches

Essentially Distributed Denial-of-Service (DDoS) attacks are flooding-based DoS attacks conducted by multiple attacking machines on the Internet. A DDoS attacker is very unlikely to have a sufficient number of computer systems to perform DDoS attacks. Instead, the set of attacking machines mostly comprises compromised computer systems and these compromised computers are so-called zombies. By aggregating the attack power of zombies, a DDoS attacker can flood Internet servers and then disable them even if they are well-equipped. In fact, DDoS attacks can inflict damages on not only Internet servers but also clients of servers being attacked. No one can actually free from the threat of DDoS attacks.

Due to the large number of vulnerable computers on the Internet and the set of easily acquired and deployed exploit programs, an attacker without much technical background can easily conduct successful DDoS attacks. Moreover, DDoS attack tools certainly will continue to evolve in order to make launching DDoS attacks even easier. For instance, techniques for scanning and compromising vulnerable computers are getting more sophisticated, and thus detection of malicious behaviors becomes much harder than before. The increasing level of sophistication also helps an attacker to identify/collect vulnerable computers more efficiently. The deployment of attack tools tends to be automatic. As a result, an attacker can widely deploy attack tools on a large set of compromised hosts with a few key strokes. All these evolutions make DDoS attacks become more powerful and harder to defend against.

In this chapter, we first present an intensive investigation on DDoS attacks, including the causes of DDoS attacks, frequently used DDoS techniques, and existing DDoS attack tools. Next, we provide an analysis on the fundamental challenges to effective DDoS defense and give an overview on existing DDoS countermeasures. Finally, a summary is given in the end of this chapter.

2.1 The DDoS Attack Problem

DDoS attacks are simply distributed versions of resource-overwhelming DoS attacks. As shown in Fig. 2.1, the major difference between a DoS attack and a DDoS attack is in the their scale – DoS attacks use one attack machine to generate attack traffic while DDoS attacks use a large numbers of attack zombies.

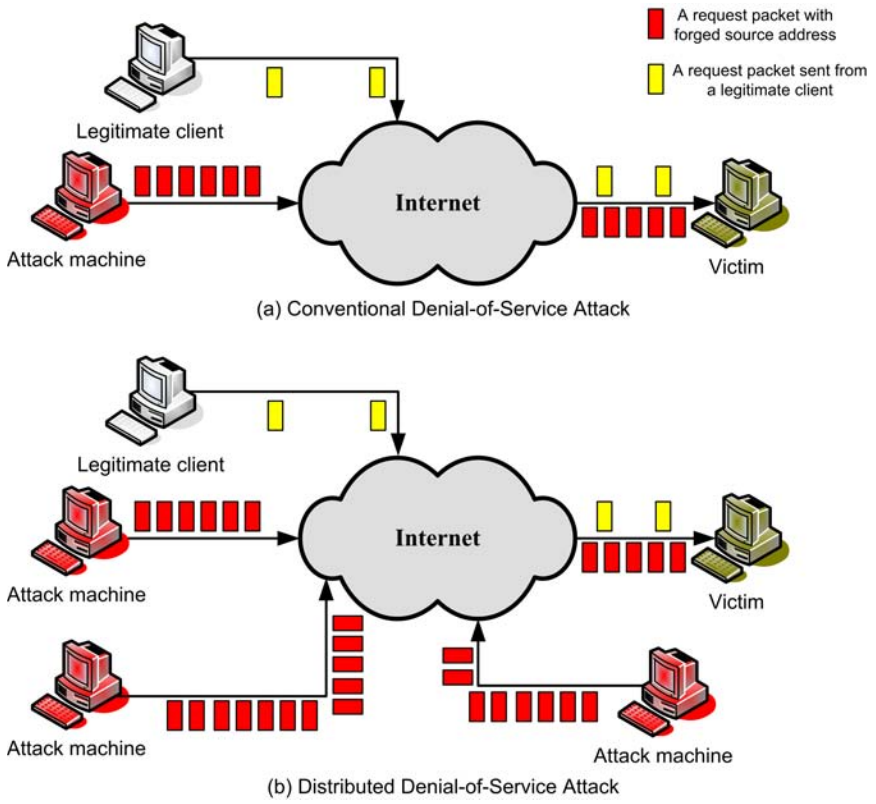

A request packet with forged source address
A request packet sent from a legitimate client
Legitimate client
Attack machine
Internet
Victim
(a) Conventional Denial-of-Service Attack
Legitimate client
Attack machine
Internet
Victim
Attack machine
Attack machine
(b) Distributed Denial-of-Service Attack

Figure 2.1: Denial of Service attacks and Distributed Denial of Service attacks.

Consider a simple DDoS attack that over-consumes bandwidth of a victim network. In this case, a vast number of large IP packets will be directed to the victim network when the attack is ongoing. Though individual IP packets toward the victim seems harmless, a multitude of them can effectively overwhelm the victim's network bandwidth or other critical packet-processing resources. Consequently requests originated from legitimate clients would not be able to compete with the malicious flood and has very little chance to acquire good service from the server. Legitimate clients would experience significant service degradation for the entire attack period. The DDoS attack streams can even effectively take the victim network off the Internet.

To turn compromised hosts into zombies, a DDoS attacker normally has to install some attack softwares on compromised systems. The execution process of these attack software are called *attack daemons*. In addition, the attacker would also install *root kits* [37, 66] on compromised systems in order to hide installed attack programs and the execution of daemon processes.

To manage compromise computer systems, in general the attacker will organize them into a hierarchical structure. As depicted in Fig. 2.2, in the top of the hierarchy is the attacker who takes control over all computers in lower layers. Hosts in the lowest layer are attack zombies, which create flooding streams and other intermediary computer systems in the hierarchy are referred to as *masters*, which are responsible for forwarding attack instructions from the attacker to zombies. Upon the receipt of attack instructions, zombies create attack streams of particular types and content at a given time, and the attack will last for a specified time duration.

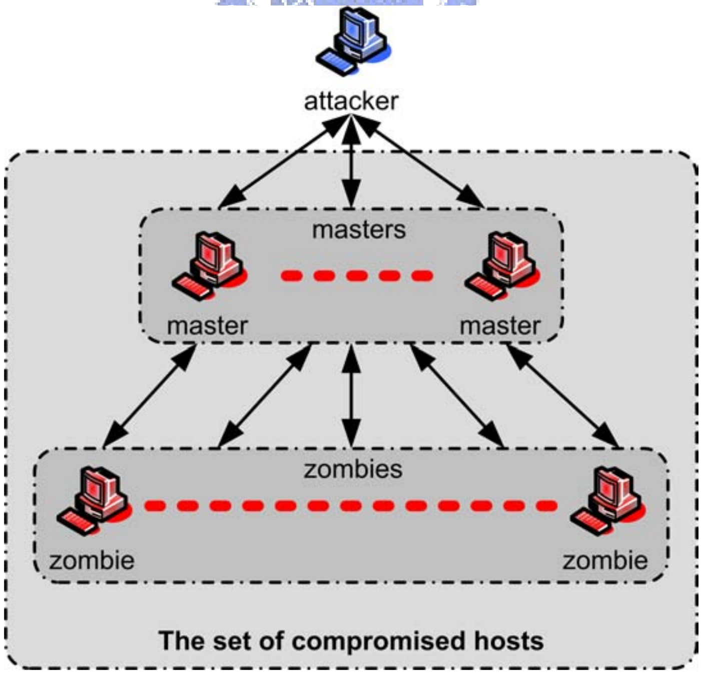


Figure 2.2: The hierarchy of compromised computer systems.

2.1.1 Direct and Indirect DDoS Attacks

According to the way DDoS attack streams are created, DDoS attacks can be categorized into two classes: *direct DDoS attacks* and *indirect DDoS attacks*. As shown in Fig. 2.3, in a direct DDoS attack, attack streams are generated directly from attack zombies, while in an indirect DDoS attack, the attacker takes advantage of other uncompromised systems, generally called *reflectors*, to attack victims. Indirect DDoS attacks are also referred to as *Distributed Reflector DoS* (DRDoS) attacks [51, 84]. In a DRDoS attack, innocent Internet servers, which will automatically reply a service request with a relatively larger (in terms of message size) response message, are selected as a candidate of a reflector. The property of generating response automatically allows attackers to further amplify the volume of attack traffic created by zombies.

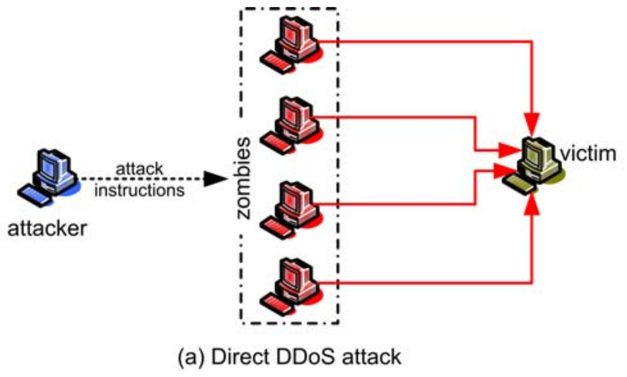
To conduct a DRDoS attack, an attacker needs to construct a list of reflectors which will later be used to reflect attack streams. Zombies are instructed to send a multitude of service requests, with victim's IP address as source IP addresses, to reflectors. And then a larger multitude of response messages will be returned to the victim. In this way, the attack traffic, in the form of normal response messages originated from selected innocent reflectors, will fill up inbound link bandwidth of the victim's network.

2.1.2 Types of Resource Exhaustion

Basically there are three types of resource exhaustion in DDoS/DRDoS attacks: network bandwidth, memory space and computation power. Each resource exhaustion can be accomplished via many different types of attack flows. For instance, network bandwidth can be exhausted by a high volume of network traffics. Memory resource can be tied up by a multitude of service requests, each of which will occupy a portion of memory space at the victim machine. And, computation power can be over-consumed by large numbers of high-computation load service requests. In this section, we will present details on overwhelming the three major types of resources.

Network Bandwidth

Fig. 2.4 illustrates bandwidth exhaustion DDoS attacks. The goal of bandwidth flooding attacks is to create severe network congestions on network links of a victim or a victim



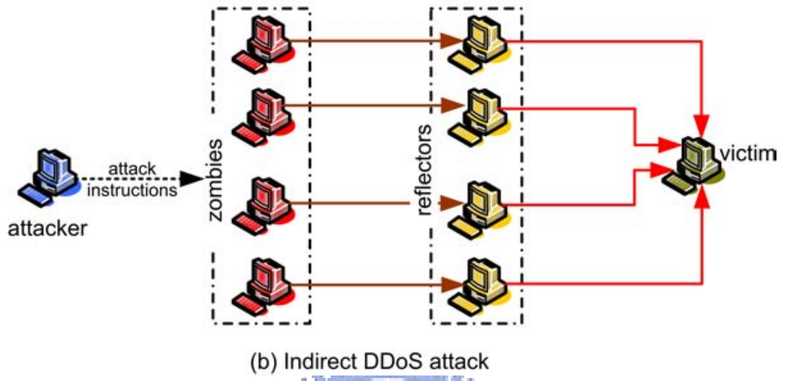


Figure 2.3: Direct and indirect flooding streams

network. To this end, zombies generally will send full-size IP packets to the victim. Especially the inbound network bandwidth will be saturated with attack packets, and as a result, a high percentage of legitimate packets would be discarded.

Smurf attack [21] is a well-known DRDoS attack that creates a high volume of attack traffic using reflectors. In a smurf attack, zombies broadcast ICMP echo requests [93] with victim IP address as source IP addresses to a local network. (That is, the destination IP address is an broadcast address, and victim's IP address is used as the source IP addresses.) Computers receiving the forged ICMP echo requests will return ICMP echo reply messages to the victim. In this example, it is clear that computers which generate ICMP echo reply messages are treated as reflectors. With a sufficient number of such reflectors, the inbound network bandwidth of victim's network can be exhausted. It is worthy to note that sometimes smurf attacks may also inflict damages on reflectors. Consider that there are n reflectors in the same local network, and each ICMP echo reply message is s-bit long. Let the attacker send f ICMP echo requests per second. Then, the outbound bandwidth of reflectors' network will also be exhausted if it is smaller than $n * s * f$.

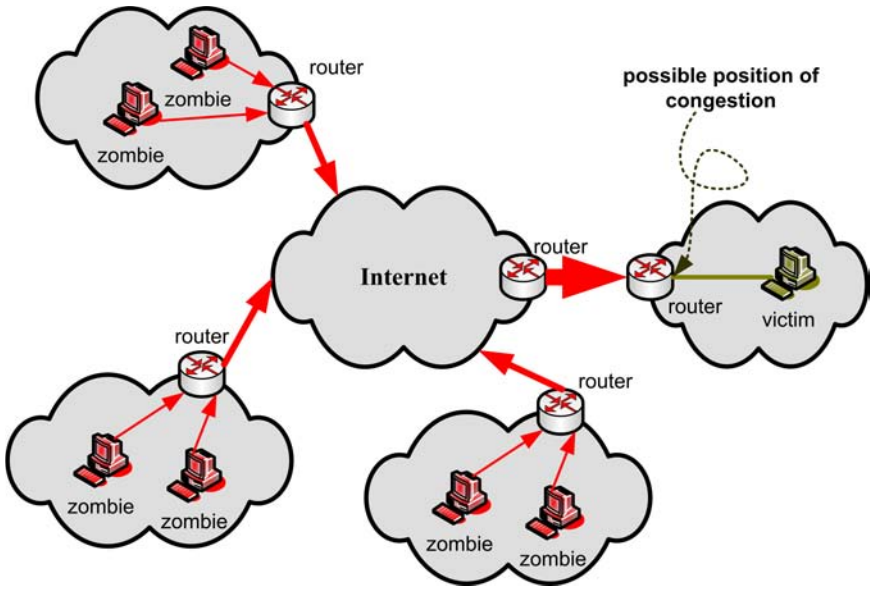


Figure 2.4: The aggregate of DDoS attack streams exhausting network bandwidth.

In addition to ICMP, attackers can employ many other protocols, such TCP, HTTP and DNS to trigger the generation of huge amount of network traffic from innocent computer systems [84]. Via raw socket interfaces [3, 109], most zombies can send TCP SYN packets, TCP SYN-ACK packets, HTTP requests and DNS requests, with forged source IP addresses to a corresponding list of reflectors (i.e. computers listening on a particular TCP port, HTTP servers and DNS servers), and consequently the victim's network would be disabled owing to bandwidth exhaustion.

There are still other ways to create bandwidth attacks. For instance, by connecting two UDP services, each of which generates outputs, we can create a huge amount of traffics transmitted between the two service machines in an infinite loop. This will result in denial of services on both service machines and on intermediary networks between them. One such example is given in [15], and it is summarized below. **Chargen** [91] is a service running on port 19. Whenever an UDP datagram is received, chargen will return an answering datagram containing a random number (between 0 and 512) of characters. **Echo** [92] service runs on port 7. When an UDP datagram is received, the data from it is sent back in an answering datagram to the original sender. An attacker can easily connect the chargen service running on one machine to the echo service on another machine by one forged echo or chargen datagram to a chargen or a echo service respectively. Once the two services are connected, the created traffic volume can persistently consume a huge amount of network bandwidth in an endless loop, causing loss of packets or exhausting the power of two machines (running echo and chargen services) to process packets that are echoed between them.

Buffer Space

Another commonly used method to penetrate denial-of-service attacks is to tie up some particular buffer space which plays a critical role in the execution of the server process. In general, this would lead to the suspension of the service for as long as the attack lasts.

TCP SYN flooding attack [16] is a well-known DDoS attack in this category. To better understand how TCP SYN flooding attack works, we need first review the establishment of a TCP connection. As shown in Fig. 2.5, the establishment of a TCP connection requires exchanging three packets between two communicating peers, and this procedure is widely known as the *TCP three-way handshake*. The system that wants to establish a TCP connection is referred to as a TCP client, and the system a TCP client connects to is called a TCP server. In TCP three-way handshake, the TCP client first sends a SYN packet expressing its attempt to connect to the TCP server. The SYN packet notifies the TCP server of the initial sequence number (ISN) used by the TCP client. Presume that the ISN recorded in the SYN packet is x, and the TCP server accepts the connection-initiating request. The TCP server acknowledges the receipt of the TCP SYN packet and sends its own ISN, y, in a SYN-ACK response packet. The SYN-ACK packet contains an ACK field with a value $x+1$. A TCP connection in this state is called a *half-established connection* and each half-established TCP connection occupies an entry, storing parameters of the pending connection (e.g., received ISN and other connection attributes), in a *half-established connection queue*. (If the connection initiating request is not accepted, a RST-ACK packet, or an ICMP port-unreachable packet, is returned to the TCP client.) Afterwards, the TCP client finishes the handshake procedure by sending an ACK packet which acknowledges the receipt of the SYN-ACK packets. The ACK field in this packet is $y+1$.

In this way, a bi-directional connection is established between the TCP client and the TCP server. The entry in the half-established connection queue, which corresponding to a newly established TCP connection is removed. Without the last ACK packet delivered from the TCP client, the data structure describing a pending connection will be maintained for a predefined period of time, and the memory space allocated for it will be deallocated when timeout.

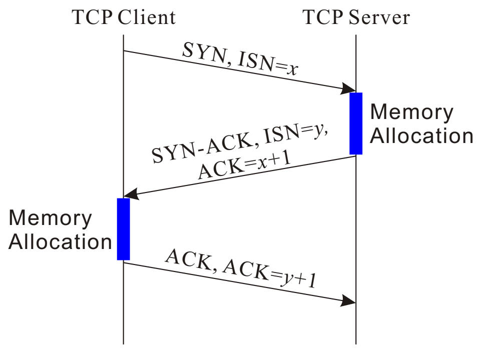


Figure 2.5: TCP three-way handshaking

The weakness of TCP three-way handshaking is on the allocation of memory space for half-established connections. This give malicious users a chance to tie up half-established connection queue using a vast number of TCP SYN packets, as depicted in Fig. 2.6. In a TCP SYN flooding attack, the source IP addresses of malicious SYN packets are forged intensionally. As a result, the last ACK packets corresponding to the malicious SYN packets will never return to the victim server. By persistently sending a large number of such malicious SYN packets to the victim, an attacker can effectively stop the victim from accepting new TCP connections. In this way, services relying upon TCP connections are disabled.

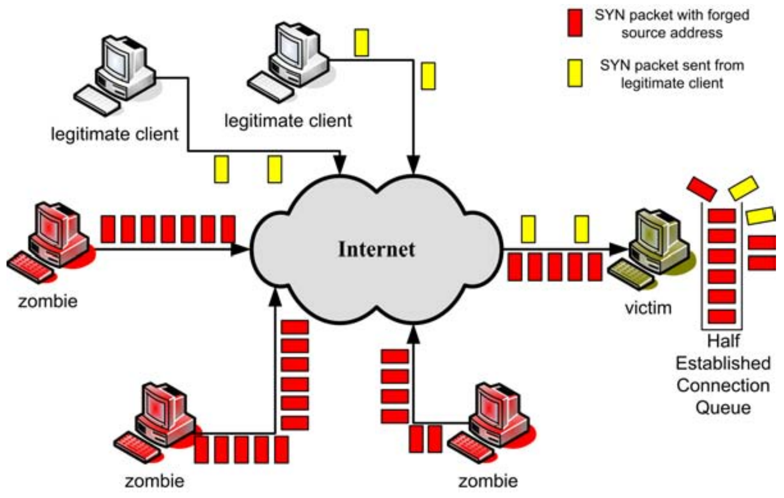

SYN packet with forged source address
SYN packet sent from legitimate client
legitimate client
legitimate client
Internet
zombie
victim
Half Established Connection Queue
zombie
zombie

Figure 2.6: Distributed TCP SYN flooding attack

Computation Power

Some other types of DDoS attacks aim at the exhaustion of computing power of the victim rather than its network bandwidth or memory space. Herein the computing power mostly refers to the capability of processing network packets or encrypting/decrypting communication contents. Services that commit extensive computations prior to the authentication of service requests are vulnerable to such kind of DDoS attacks. For instance, firewalls, intrusion detection systems and Internet routers have to commit some of their computing capabilities to each received packets. This characteristic might encourage malicious users to tie up the processing power of these network entities by feeding them a large number of small packets in a burst. Though such attack traffic may only occupy a small portion of available link bandwidth, an attacker can keep Internet routers/firewalls/IDSs very busy in processing a flood of small packets and legitimate packets may be discarded owning to the lack of enough power to process them.

Furthermore, attackers can take advantage of some improperly designed security protocols, especially authentication protocols, to exhaust the CPU resource of the victim. This is mainly because many authentication protocols are based on public-key cryptography in which computationally-expensive modular exponentiation operations are extensively used. If a service involves authenticating the validity of a service request, using public-key cryptography, right after the request arrives at, the service is subject to computation power based DDoS attacks. That is, an attacker can initiate a such kind of DDoS attack by sending the server a lot of bogus service request. Consequently the computation power of the victim server will be exhausted in order to honestly authenticate all these malicious requests.

2.2 DDoS Attack Tools

It has been made very clear that, to conduct DDoS attacks, an attacker first needs to take control over a large number of zombies. An attacker uses exploit programs to penetrate vulnerable computer systems and then installs attack programs on them. After that, the attacker can conduct DDoS attacks by sending zombies attack commands asking them to flood victims with specified attack streams.

In order to collect zombies rapidly, in general, attackers will use network exploration tools and security assessment softwares, such as nmap [50] and nessus [39], to scan a extremely large number of systems. Both nmap and nessus are capable of detecting vulnerabilities on target machines. Particularly nmap was designed to identify available hosts on the Internet and to recognize operating systems and Internet services running on those hosts. It can also detect firewalls deployed in the network being scanned. By cooperating with nmap, nessus is capable of discovering known vulnerabilities on target machines automatically. Both systems were originally designed to help system administrators identify security flaws on machines in their administrative domain. Unfortunately from the dark side, both systems can also help attackers find weaknesses of computer systems and subsequently allow attackers to penetrate vulnerable systems.

In addition to the infection process, the attack softwares running on masters and zombies are another core of DDoS attacks. Most of DDoS attack programs are built upon code fragments adopted from other DDoS attack tools. In other words, DDoS attack tools are enhanced incrementally, and they are mostly differ in the communication mechanisms between attacker and masters, and between masters and zombies. Moreover, DDoS attack tools could provide different customizations in creating attack flows. In the following, we will summarize characteristics of known DDoS attack tools according to some reports on them [8, 22–24, 26, 41–45].

Trinoo

Trinoo [23, 43] is a DDoS attack tool which creates UDP flooding attacks. The attacker sends attack commands to masters via TCP, and masters communicate with zombies via UDP. According to attack instructions, trinoo zombies will flood victims using UDP packets of a given size, over random UDP ports and for a specified attack duration. Notice that the source IP addresses of the UDP flooding packets are not spoofed. Both masters and zombies are protected by password, i.e. communications to masters or zombies require a password. Moreover, if a second connection is made to a master (or zombie) while the attacker is connected, an alert message containing the IP address of the second connection will be delivered to the attacker.

The trinoo attack program (i.e. the executable of the trinoo daemon process) contains IP addresses of trinoo masters. Upon the execution of a trinoo daemon on a zombie, it will announce its availability to one of the default trinoo masters by sending the default master a UDP message containing the string "*HELLO*". The trinoo master then stores a list of known zombies in a file named "..." in an encrypted form. To check the status of those zombies, a trinoo master broadcasts a request to all known daemons, and daemons receiving the request will return an UDP packet containing the string "PONG".

Tribe Flood Network

Tribe Flood Network (TFN) [23, 45] employs a master and zombie architecture which is similar to that of trinoo. TFN is capable of launching UDP flooding, TCP SYN flooding, ICMP echo flooding and smurf attacks at specific or random ports. It is worthy to note that TFN can generate attack streams with spoofed source IP addresses, and sizes of attack packets can be altered.

A TFN master accepts attack instructions through command line interface. This is generally done via normal telnet sessions, SSH sessions or remote shell bound to a specific TCP port. Each TFN master maintains a list of known zombie which is supplied by the attacker. It was reported that some TFN masters may encrypt the zombie list using blowfish algorithm [99]. A TFN master communicates with zombies using ICMP echo reply packets. An attack instruction is encoded as a 16-bit binary value, which is defined at the compile time of the master executable. The 16-bit attack instruction is stored in the ID field of an ICMP echo reply packet, and additional arguments are embedded in the data portion of the packet.

Stacheldraht

Stacheldraht [26, 44] is a DDoS attack tool that adopts features from trinoo and TFN. Communications between an attacker and masters are protected using blowfish encryption algorithm. Furthermore the executable of daemon process can be updated automatically on demand. The attacker uses a telnet-like session to communicate with masters, and masters send attack commands to zombies via ICMP echo reply packets. Finally, stacheldraht can launch UDP flooding, TCP SYN flooding, ICMP echo flooding, and smurf attacks.

Shaft

Shaft [24, 41, 42] is a DDoS tool which creates UDP flooding, TCP SYN flooding, TCP ACK flooding, ICMP flooding or a mixture of them. The source ports and source IP addresses of shaft attack packets are randomized. The attacker communicates with masters via telnet connections. Communications between masters and zombies are accomplished via UDP packets.

Furthermore a “ticket” mechanism is developed to keep track of individual zombies. Whenever a shaft zombie starts up, it announces its availability to its default master by sending a “new” command and a password. Upon receipt of the announcement, the corresponding master returns a ticket number to the zombie. Afterwards, both the password and the ticket number must have a match for a zombie to execute an attack command. Finally, the creators of shaft attack tool are particularly interested in packet statistics. A shaft master can query zombies for obtaining the amount of malicious traffic generated. This might be used to estimate the amount of traffic created by shaft zombies.

TFN2k

As its name indicates, TFN2k [8,22] is a successor of TFN. It employs UDP flooding, TCP SYN flooding, ICMP flooding, smurf attacks or a mixture of them. It can also obfuscates the traffic sources by forging source IP addresses randomly or within a specific range of IP addresses. This allows TFN attack packets circumventing ingress filters [49]. (Ingress filtering will be described later in this chapter.) In addition, TFN2k is capable of creating random mal-formatted IP packets which can crash some IP protocol stacks. Moreover, TFN2k was designed to compile on different operating systems, such as Linux, Solaris and Windows platforms. In other words, these systems are susceptible to be infected by TFN2k.

Communications between an TFN2k attacker and masters are accomplished via randomized TCP, UDP and ICMP packets, and is protected by encryption algorithms. Masters and zombies are protected by passwords determined at compile time. All attack commands are represented using a single character, and therefore it is very difficult to identify packets carrying TFN2k attack instructions by inspecting payloads of IP packets. Furthermore TFN2k attackers can hide the origins of attack instructions by IP spoofing. As a consequence, there is no way for acknowledging the receipt of attack commands. To increase the probability that masters can receive attack instructions successfully, attack instructions are issued multiple times. All these factors make TFN2k very hard to defend against.

Mstream

Like other DDOS attack tools, mstream [25,36] conducts DDoS attacks via masters and dozens of zombies. Communications between an attacker and masters are accomplished via TCP, and communications between masters and zombies are through UDP. Compared with other DDoS attack tools, mstream is relatively primitive. It creates a flood of TCP packets, with ACK bit set, and the source IP addresses of these attack packets are randomly spoofed. A vast number of attack packets consume inbound network bandwidth of the victim network. Additionally, the TCP RST packets triggered by the attack packets would also tie up outbound network bandwidth of the victim network.

Trinity and Entitee

Trinity and its variant, Entitee, [55], are DDoS attack tools using IRC to deliver attack instructions. Upon the execution of attack daemons, zombies will connect to an IRC server and then wait for commands sent from the attacker. Using IRC, an attack instruction can be delivered to a single trinity zombie individually, or it can be broadcasted to a specific IRC channel. In the latter case, all zombie receiving the instructions will execute the attack command accordingly. Available DDoS attacks are UDP flooding, IP fragment, TCP SYN flooding, TCP RST flooding [108] and TCP ACK flooding [25,36].

2.3 Fundamental Challenges to DDoS Defense

DDoS attacks are rooted on the resource asymmetry between zombies and the victim under attack. They are, in fact, simple brute force attacks. It is the simplicity makes DDoS attacks very hard to defend against. Additionally, there are other factors that constitute obstacles to the design of effective DDoS countermeasures, and major obstacles are described below.

- **IP header fields can be spoofed.** As above-mentioned, most DDoS attack tools are capable of forging arbitrary IP header fields, such as source IP address, source/destination ports and other header fields. Especially they can create IP packets with source IP addresses being spoofed randomly or within a specific range of IP addresses. IP spoofing allows attackers *canceling the sources of attack streams or attack instructions*. Thus, attackers run at low risk of being caught because the source IP addresses are often implausible. It is very hard for victims to trace back to zombies, and it is even harder to find the attackers controlling zombies and masters. Furthermore, IP spoofing helps attackers to circumvent sophisticate queuing algorithm [2, 62], which help share limited resource fair among several participants. It is also worthy to note that IP spoofing is a key to create DRDoS attacks. In DRDoS attacks, zombies use IP spoofing to deceive selected Internet servers into flooding victims.

- **Attack streams are composed of legitimate-like packets.** To avoid being detected and filtered by intrusion detection systems and firewalls, DDoS attack tools tend to use common or expected communication protocols as vehicles of their attack streams. Most DDoS attack tools use TCP, UDP, ICMP, upper layer protocols or a mixture of them to flood the victim. For instance, attackers can use a huge number of large-size HTTP requests to saturate the network bandwidth of a web server or to overwhelm the web server. IDSs that use explicit attack patterns to detect attack packets cannot work well in this situation. Furthermore legitimate packets could suffer from collateral damages. In other words, filtering attack packets in an effective and accurate manner remains a very challenging issue.

- **Computer systems are usually exploitable.** One simple solution against DDoS attacks is to secure all computer systems on the Internet. In this way, it will be hard for attackers to take control over a large number of zombies, and thus conducting DDoS attacks would become harder. Unfortunately, this simple approach is infeasible. First, computer systems on the Internet are administrated independently. It requires tremendous efforts to ensure all computer systems in different administrative domains from being compromised. Second, securing a computer system itself is a very difficult problem. Computer systems can be vulnerable for various reasons. For instance, an attacker can penetrate a system by taking advantage of software bugs, mis-configurations or weak passwords. All these problems together make securing a computer systems a very complicated issue, and it is even harder to assure the security of all computers. In other words, it is unrealistic to assume that we can secure all computers on the Internet. There will always be abundant attack zombies for DDoS attackers.

- **No one-size-fit-all solution for DDoS attacks.** The diversity of DDoS attacks, especially in terms of types of resource exhaustion, further complicates the design of effective DDoS countermeasures. Some DDoS attacks tend to be detected easier on victims or victims' networks than in other positions, while others DDoS attacks need to be handled by Internet routers in a cooperative manner. Specifically, attack detection can be very effective at victims or victim networks, while attack filtering is usually ineffective in these places. On the other hand, attack filterig can be effective at upstream of the attack, while it is usually also very hard to detect attack upstream.

2.4 An Overview of DDoS Countermeasures

In recent years, many DDoS defense mechanisms have been presented in the literature. Some of these approaches attempted to eliminate DDoS attacks by removing fundamental causes of attacks, and such DDoS solutions are usually referred to as *proactive defense*. Some approaches, classified as *reactive defense*, aim at detecting DDoS attacks and at alleviating the syndrome of attacks when attacks are ongoing. And some other DDoS defense systems focus on locating the sources of attack streams. With IP traceback mechanisms, attacker would run higher risk of being discovered, and this may help stop future attacks.

In this section, we first present several candidate positions for the deployment of DDoS defense systems. Afterwards, we briefly describe design concepts and operational overview of existing proactive defense approaches, reactive defense approaches and traceback schemes.

2.4.1 Points of Deployment

Fig. 2.7 depicts candidate points for the deployment of DDoS defense systems. Basically, DDoS defense can be placed at *victim-end elements*, *source-end elements*, or multiple defense nodes, including victim-end elements, intermediate network components and

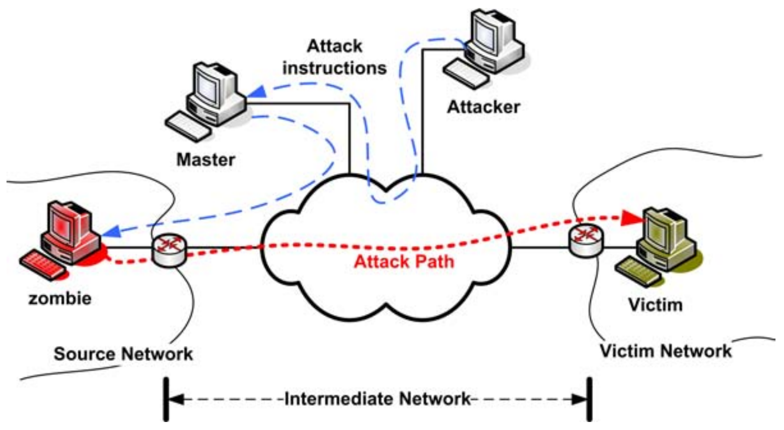


Figure 2.7: Candidate points of deployment

source-end elements. In the context of this dissertation, victim-end elements include end host systems and edge routers which serve as ingress and egress points for subnetworks. Intermediate network components mainly direct to Internet routers. Source-end elements, much like victim-end elements, comprise also end host systems and edge routers. Though victim-end elements and source-end elements both refer to almost the same set of network entities, the functions of DDoS defense systems deploy on victim-end elements and source-end elements are very different, and this will be make clear later in this section.

First, consider a victim-end deployment for DDoS defense systems. DDoS defense systems in this category protect an end hosts if the defenses are installed on an end host system. Or they protect a set of end hosts connecting to an edge router if the defense are deployed on an edge router. Owing to the direct benefits to the deployed sites, victim-end approaches generally have a stronger incentive to be widely deployed. Since DDoS attack traffic aggregates in points approaching victims or victim networks, traffic anomalies will be more observable on victim-end elements. However, victim-end defenses sometimes provide little help in filtering attack packets if the competition for resources take places on upstreams routers.

Second, DDoS defense systems can be deployed on end host systems or edge routers in order to prevent deployed sites or subnetworks from participating in DDoS attacks. The major advantage of such deployment is the ability to confine attack traffics at their sources or sources networks. DDoS attacks can be suppressed quickly, and network congestions caused by attack traffic can be reduced. Though schemes in this category seems very effective in defending against DDoS attacks, they need to be deployed widely on the Internet, and they generally provide very little motivations for their deployment.

Next, consider a distributed deployment of DDoS defense. That is, DDoS defense systems are deployed on multiple nodes including end hosts, edge routers and Internet core routers. In this case, the DDoS problem is addressed in a cooperative manner. Whenever a DDoS attack signal is detected on victims or edge routers, an alert is send to upstream defense nodes. In this way, we could push the defense line approaching attack sources, and the filtering components may achieve better performances. The effectiveness of distributed deployment largely depends on tight cooperation of defense nodes. These nodes may belong to different autonomous systems that are administrated independently, and such a distributed deployment could incur tremendous coordination efforts among different ISP networks. Furthermore, alert messages containing packet filtering instructions need to be protected. Especially they should be authenticated; otherwise, malicious users can instruct defense entities which are capable of filtering packets to arbitrarily discard legitimate packets. The establishment of secure communication channels among defense entities would also incur substantial cost.

2.4.2 Proactive Defense Approaches

One naive approach to improve a server's resistance to DDoS attacks is to allocate resources redundantly. By replicating critical resources, the difficulty of conducting a DDOS attack on the server is arisen. This naive approach might work in defending against a DoS attack or a small-scale DDoS attack. However, it certainly cannot cope with DDoS attacks with a large number of zombies.

Another simple proactive approach is to secure all computers systems on the Internet. By minimizing the candidates of attack zombies, we may have a chance to weaken the power of DDoS attacks at some level, and resource-replicating approaches might work better. However, as aforementioned, this approach is infeasible on large scale networks, such as the Internet, and in other words, though approaches described above might be helpful, they simply cannot stop DDoS attacks.

A well-known proactive scheme for addressing DDoS attacks is so-called *ingress filtering* [49, 70]. Ingress filtering is a source-end defense system. Assume that an edge router has the set of valid IP addresses used by computer systems connecting to it. Then an ingress filter, installed on an edge router, is capable of prohibiting attackers within the originating network from launching DDoS attacks with forged source IP addresses that do not conform to valid IP addresses. In this way, ingress filters installed at the periphery networks can help reduce the power of spoofed DDoS attacks. Another similar approach, named DPF, is proposed in [82]. DPF addressed DDoS attacks by preventing spoofed IP packets from reaching their destinations. In DPF, given the reachability constraints imposed by routing topology, a border gateway can determine whether an incoming IP packet is valid or not by inspecting its inscribed source/destination IP addresses. DPF requires only 18% of deployment in Internet AS topologies to achieve a synergistic filtering effect. However, its ultimate effect is identical to that of a widely deploy ingress filtering scheme.

Secure Overlay Service (SOS) [60,61] is proposed to protect a predefined set of Internet servers and clients from DDoS attacks. In brief, the SOS employs a set of overlay nodes, secure overlay tunneling, and filtering to control access to protected sites. SOS works well under simple congestion based DDoS attacks, but it is vulnerable under two more intelligent attacks [113]. A mechanism for protecting web servers from DDoS attacks, called WebSOS [33], is built on the SOS architecture and the use of TLS/SSL [40, 94]. WebSOS is simply a direct extension of SOS. That is, it allows a predetermined set of clients scattered on the Internet to access a protected web site.

In [112], a DDoS defense scheme for sustaining availability of web servers is presented. Protection is simply achieved by redirecting clients to a new IP address and port number via a standard HTTP redirection message. Part of the new IP address and port number will be translated into a Message Authentication Code (MAC) for the client. If the source IP address of the first HTTP request is forged, the attacker would not received the redirection message. Thus without a valid MAC, attacker will not be able to arbitrarily access a protected web server and therefore the high availability of the server can be sustained.

An IP-layer anonymizing infrastructure, called ANON [29,67,68], is proposed to hide actual IP addresses of Internet servers from legitimate clients and attackers, while at the same time, the services are sustained. With ANON, a client communicates with its target server by using "server handles" rather than server IP address, where a server handle is actually an encrypted string from which can be decrypted into the server IP address. Other techniques, such as protocol camouflaging, link encryption, link padding are also adopted to protect against attackers from learning the IP addresses that ANON intend to hide. In [63], a hardware DDoS solutions that can block malicious TCP packets at edge routers is proposed. Simply speaking, this approach determine the legitimacy of TCP packet by examining whether the packets belongs to a previously established TCP connection or not. TCP packets, except for TCP SYN packets, that do not in any known TCP connections are discarded.

2.4.3 Reactive Defense Approaches

Next, we turn to look at reactive approaches which respond to DDoS attacks when they occur. This kind of DDoS defense approaches generally consist of two phases: *attack detection phase* and *attack filtering phase*. The mission of attack detection phase is to determine whether a DDoS attack is ongoing or not. Upon the recognition of DDoS attacks at the detection phase, components responsible for attack filtering start to distinguish malicious packets from legal ones. Identified attack packets will be dropped, or rate-limiting rules will be imposed on identified attack flows. In the following, we will group reactive defense approaches according to their deployment, and present an overview on each schemes.

Victim-end DDoS defense

A preliminary study on detecting and responding to DDoS attack with chi-square statistic and entropy is presented in [47, 48]. Techniques for DDoS detection and response is briefly introduced, while the effectiveness and reliability of the proposed approach is not evaluated. There are still some other schemes detecting DDoS attacks with statistical techniques [72, 111]. These schemes mostly focus on detecting attacks, and responding techniques are seldom discussed.

PacketScore [31, 64] is another scheme using statistical techniques to detect DDoS attacks. In PacketScore, profiles of incoming traffic are compared with normal traffic profiles in order to detect anomalies. Given the attributes of an incoming IP packets (i.e. IP header fields), PacketScore system computes "Conditional Legitimate Probability" (CLP) of the incoming packet, and by comparing the CLP of each packet with a dynamically adjusted threshold, attack packets can be identified and discarded selectively.

NetBouncer [106] is a victim-end solution to DDoS attacks. Upon the receipt of an incoming packet, a NetBouncer device determines the legitimacy of the packet by challenging the sender. Several techniques for client-legitimacy test are discussed. HIF [85,87] is also a victim-end DDoS defense. It detects DDoS attacks by monitoring the source IP addresses of incoming packets. A surge in the number of previously unobserved source IP addresses is considered a signal of DDoS attacks. Subsequently, packets with unobserved source IP addresses are discarded. HCF [57] uses the TTL information inscribed in each IP packets to identified spoofed packets. Since spoofed packets generally do not traversed the Internet path between the victim and the IP address being spoofed, spoofed packets will arrive at the victim with an incorrect TTL value. In this way, spoofed packets, which constituting a majority of DDoS packets, can be identified and then discarded.

Distributed DDoS defense

Owing to the distributed nature of DDoS attacks, many researchers choose to deploy DDoS defense nodes distributed on the Internet. For instance, in [107], a global defense infrastructure (GDI) is proposed. A GDI is actually a network of local detection systems (LDSes) distributed on the Internet. A LDS is responsible for detecting suspicious DDoS flows, and then determines if a DDoS attack is ongoing according to local detection information and remote detection information obtained from other LDSes. Once a DDoS attack is confirmed, LDSes will instruct routers to block attack streams. This work mainly focus on the design of the infrastructure and placement of LDSes. Few technical details on detecting DDoS attack flows are involved.

Much like GDI, a coordinated DDoS detection and suppression architecture, named COSSACK, is presented in [81]. A major component in COSSACK is a software element, called watchdog, deployed at edge routers. It monitors inbound and outbound traffic of its own network and exchange information with other watchdogs. Once the watchdog deployed near the victim notices a signal of DDoS attacks, it notifies other watchdogs deployed at source networks of attack streams. Subsequently each source network watchdogs inspect their outgoing flows to see if zombies exist in their monitored networks. Watchdogs that identify zombies can then take responsive action to stop the attack.

Pushback [56, 73] is a cooperative DDoS defense mechanism for controlling high network bandwidth aggregate upstream. Simply, in Pushback, a congested router is allowed to send rate-limiting requests to contributing neighbors (i.e. adjacent routers sending a significant fraction of traffic). A receiving router can recursively propagate this rate-limiting request further upstream if it is also congested owing to the rate-limiting effect.

AITF [5,6] also addresses the DDoS problem in a cooperative manner. AITF assumes that a public-access site can detect the presence of DDoS attacks when under attack. It also assume that, with anti-spoofing mechanisms such as aforementioned ingress filtering and DPF scheme, a public-access site can identify a border gateway supporting AITF that is closest to source of an attack stream. With such massive assumptions, upon sensing DDoS attacks, the victim sends a filtering request to its local gateway. The victim's gateway then instructs the gateway which is considered closest to the attack source to block attack flows.

PI [115], similar with HCF, focus on identifying spoofed DDoS packets. In PI, Internet routers inscribe one or two bits into the IP Identification field. As a result, IP packets traversing the same Internet path will carry an identical path identifier in their Identification field. IP packets coming from different Internet paths are unlikely to have the same path identifier. In this way, PI enables victim to distinguish legitimate packets from legitimate ones.

Instead of identifying spoofed IP packets, SIFF [114] allows constructing a privileged communication channel between two communication parties. A three-way handshake protocol is developed to establish privileged channels. After a three-way handshake protocol, a legitimate clients can acquire a "capability" to insert into its privileged packets. An IP packet without capability will be treated as unprivileged, and a packet with forged capability will be discarded. In this way, SIFF helps Internet servers to maintain their availability to their clients.

In [103, 104] an IP traceback method is employed to construct the attack graph, and subsequently IP packets marked with one of network edges in the attack graph are discarded This scheme suffers from the large number of packets required to construct the attack graph. And, it may mis-classify legitimate packets as attack packets if legitimate packets ever traversed the network edge in the attack graph.

Source-end DDoS defense

To block attack streams in the first place, several schemes were proposed to detect and stop attacks at their sources. One such kind of approach, called Source Router Preferential Dropping (SRPD), is proposed in [46]. SRPD monitors high-rate outgoing flows, and if the response time of a remote network exceeds a predefined threshold, packets belonging to the high-rate flows destined to that remote network will be dropped probabilistically. Another source-end DDoS defense system, called D-WARD, is presented in [76]. D-WARD monitors various types of protocols. With a set of predefined thresholds in packet sending rate or the ratio of incoming and outgoing packets, D-WARD can detect DDoS attacks which use TCP, UDP or ICMP protocols as their vehicles. Once a DDoS attack is detected, D-WARD will impose rate-limiting rules on suspicious flows in order to mitigate the effect of detected DDoS attack.

2.4.4 Attack Traceback Approaches

Recently many research efforts have been directed to attack traceback. The term *attack traceback* refers to both the problem and the solutions of identifying Internet paths traversed by attack flows. It is believed that locating Internet paths of attack streams would subsequently help stop DDoS attacks in an automatic way.

Traceback techniques presented in the literature can be classified into three main categories: *packet marking* , *messaging*, and *hop-by-hop tracing*. As shown in Fig. 2.8, Internet routers deterministically or probabilistically embed path information into packets when they are in transit [1, 69, 86, 97, 98, 102]. The recipient of marked packets can then use embedded path information to reconstruct attack paths.

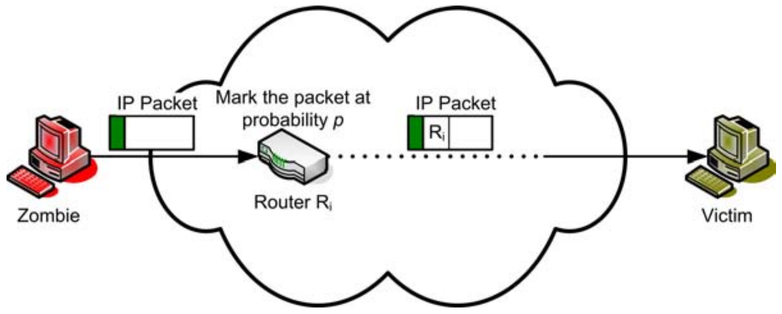


Figure 2.8: Packet marking schemes

Probabilistic Packet Marking (PPM) was originally proposed by Savage et. al. [97,98] to encode partial path information and embed encoded result into IP packets. After a sufficient large number of packets carrying partial path information are collected, a victim can identify Internet paths traversed by detected attack flows. By assuming that a victim can have a complete topological map of its upstream routers and the use of hash chains for authenticated routers, Song and Perrig [102] improve the effectiveness of PPM, in terms of accuracy and number of packets needed for path reconstruction. To further reduce the number of packets needed for path reconstruction, techniques for adjusting marking probability are studied in [86]. And, the effectiveness of PPM under various parameter settings is analyzed in [1, 69].

Deterministic Packet Marking (DPM) [9, 10] also employs packet marking techniques to identify attack paths. Instead of marking packets with a probabilistic method, in DPM, IP packets are marked deterministically when entering the network. Each edge routers are responsible for assuring the source of IP packets by marking a segment of its 32-bit IP address in IP packets. In this way, downstream routers need not participate in the packet marking process, and the number of packets needed for path construction is reduced.

An alternative way for IP traceback, called messaging, is to have Internet router transmit additional packets carrying path information to the destination. As shown in Fig. 2.9 whenever an IP packet arrives, the receiving router determines a probability (typically 1/20000) to send an addition ICMP message, containing the IP address of the forwarding router and some content of transmitted IP packet, to the destination. The main idea of this approach is that a sufficient number of DDoS attack packets will trigger the generation of such ICMP messages, and therefore a victims can reconstruct attack paths after receiving enough such ICMP packets.

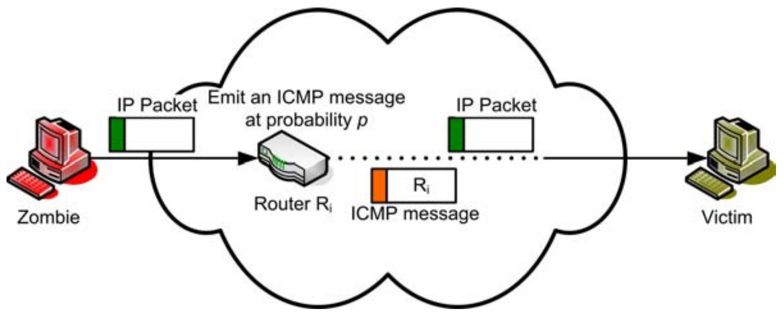


Figure 2.9: Messaging schemes

Hop-by-hop tracing refers to approaches recursively identifying routers on attack paths from the victim to the source of attack streams. The simplest hop-by-hop tracing is to manually log onto routers nearest the victim and decide next upstream routers on the attack path. Sometimes this approach is also referred to as *link testing*. By repeating this procedure, we can reach the sources of DDoS attack streams. Its main drawback is that substantial coordination efforts is needed, and it must be performed before DDoS attack traffic vanishes.

Some other schemes support attack traceback in a hop-by-hop manner by using auditing logs. As depicted in Fig. 2.10, Internet routers store audit logs of forwarded IP packets. In this way, attack paths can still be identified after zombies finish pumping out attack packets.

In [7], routers store some of IP packet fields, parts of IP payload and incoming interface for all forwarded IP packets. Then given some attack packets, sequences of routers, from the edge router of the victim to that of attack sources, can be identified recursively by using audit logs stored on Internet routers. To reduce storage requirement, SPIE [96,100] chooses to deterministically store only packet digests in Bloom filters [13]. Whenever an attack packets is detected at the victim, digests of the identified attack packet are then send to upstream routers of victim's network. If the digest of the attack packet is found in a upstream routers, the digest is then send to adjacency routers upstream. By repeating this procedure, we can identify an attack path of an individual packet.

Another approach employing bloom filter for addressing IP traceback, called ORMS, is presented in [71]. ORMS stores packet digests in a probabilistic manner. In this way, space requirement can be significantly reduced. This make ORMS applicable for logging packet digests at high link speeds. However, it is worthy to note that ORMS can only trace attack flows with a sufficient number packets.

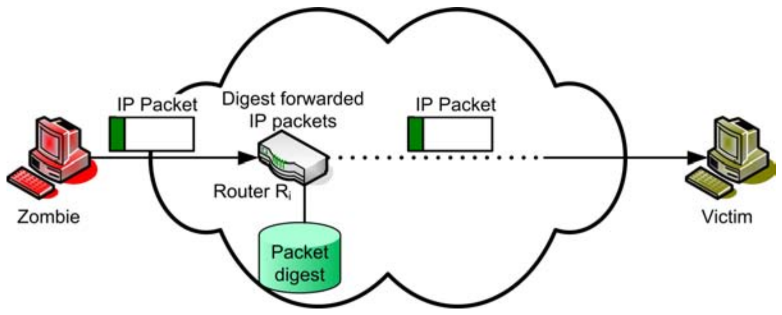


Figure 2.10: Packet digesting schemes

2.5 Summary

Distributed denial-of-service attacks are, in fact, simple resource exhaustion attacks. Though they are simple, they are also very effective. By gathering a huge amount of resources from zombies, a DDoS attacker can disable Internet services or networks with brute force. DDoS attack streams usually comprise legitimate-like packets. The presence of IP spoofing further makes DDoS attacks even harder to defend against.

In the first part of this chapter, we have briefly introduced fundamental causes of DDoS attacks, followed by a classification of DDoS attack flows, types of resource exhaustion and DDoS attack tools. Investigation presented here helps understand DDoS attacks better and supports designing appropriate and effective DDoS defense and response.

Next, we turned our attention to existing DDoS defense systems. Major obstacles to effective DDoS defense are identified. Deployment of defense mechanisms is discussed. Afterwards, we classified existing DDoS defense systems into three major groups, and then elaborated key concepts and technical details of defense approaches in each group. Points in flavor of or against these solutions are also given.

In one word, this chapter provides necessary background knowledge for study on DDoS attacks and their countermeasures. It serves a useful foundation for better understanding DDoS attacks, and would help readers to better comprehend our DDoS defense mechanism presented in the following chapters.

Chapter 3

Defending Against Spoofed DDoS Attacks with Path Fingerprint

In this chapter, we propose a new scheme, called ANTID, for detecting and filtering DDoS attacks which use spoofed packets to circumvent conventional intrusion detection schemes. The proposed anti-DDoS scheme intends to complement, rather than replace conventional schemes. By embedding in each IP packet a unique *path fingerprint* that represents the route an IP packet has traversed, ANTID is able to distinguish IP packets that traverse different Internet paths. In ANTID, a server maintains for each of its communicating clients the mapping from the client's IP address to the corresponding path fingerprint. The construction and renewal of these mappings is performed in an on-demand fashion that helps reduce the cost of maintenance. With presence of the mapping table, the onset of a spoofed DDoS attack can be detected by observing a surge of spoofed packets. Consequently, spoofed attack packets are filtered so as to sustain the quality of protected Internet services. ANTID is lightweight, robust, and incrementally deployable. Our experiment results showed that the proposed scheme can detect 99.95% spoofed IP packets and can discard them with little collateral damage to legitimate clients. It also showed that the higher the aggregated attack rate is, the sooner the attack can be detected.

3.1 Preliminaries

As aforementioned, DDoS attacks have posed a major threat to the availability of Internet services. A DDoS attacker can greatly reduce the quality of a target Internet service or even can completely break the network connectivity of a server by persistently overloading critical network or system resources of the target, such as network bandwidth, router processing capability, or CPU/memory at the target machine. Due to the lack of built-in security mechanisms in the current Internet infrastructure, conducting a DDoS attack is easy. On the other hand, defending against DDoS attacks is extremely difficult because there is usually no explicit attack patterns to distinguish legitimate packets from malicious ones. Moreover, to hide the sources of attack traffic and circumvent DDoS defense mechanisms relying on inspecting IP header fields, DDoS attack programs generally fill IP header fields, especially the 32-bit source IP address, with randomized values. This IP spoofing technique has made the detection and filtering of DDoS traffic extremely difficult, and it has become a common feature of the many DDoS attack tools.

To defend against DDoS attacks, many countermeasures have been proposed in the literature in recent years and have been reviewed in Chapter 2. Most of these schemes [9, 12, 38, 56, 60, 61, 67, 68, 70, 73, 73, 76, 96–98, 100–102] are somewhat weak in defending against DDoS attacks in large networks, such as the Internet. Some of them require supports from all edge routers and complex cooperation among different ISP networks, while others do not provide real time response to an attack. Non-trivial packet filtering operations are usually involved in the process of packet forwarding. Thus, the throughput of routers, which participate in the defense of DDoS attacks, will be significantly reduced. Other schemes [57, 85, 87, 103, 104, 115] suffer from their inherent design weaknesses, such as being vulnerable to sophisticated DDoS attacks, unable to distinguish DDoS attacks from flash crowd events, or ineffectiveness under spoofed DDoS attacks. For instance, in [57], spoofed DDoS packets are identified according to the hop-count information, which refers to the number of routers traversed in an Internet path. The effectiveness of such hop-count filtering is based on an assumption that most spoofed IP packets do not have hop-count values identical to those of IP addresses being spoofed. By inferring the hop-count information from the TTL field, spoofed IP packets can be easily identified and then be discarded when the victim is under attack. However, this scheme has problems in its basic assumption, that is, a DDoS attacker can obtain the hop-count value between the victim and a spoofed IP address. As we will show in Section 3.2, acquiring the hop-count information requires only two *traceroute* operations. Thus, hop-count filtering is likely to be defected by sophisticated DDoS attacks that can adjust the initial TTL value according to traceroute results.

Based on the 32-bit source IP address, an IP filtering technique for defending against DDoS attacks is proposed [85, 87]. In this approach, the number of new IP addresses connecting to the protected server is monitored, and IP addresses of frequently contacted clients are learned from past communications. A surge in the number of new IP addresses is considered as a signal of the onset of a spoofed DDoS attack. Then, packets with source IP addresses not found in the IP address database will be discarded during the attack. This approach suffers from several drawbacks. First, it cannot distinguish flash crowd events from real spoofed DDoS attacks. Second, before launching a real DDoS attack, an attacker can slowly pollute the IP address database of the victim by sending the victim a set of malicious packets with an IP address to be spoofed. Then, the attacker can use those IP addresses used before to attack the target. Although it is indicated [85] that such an attack can be prevented by increasing the period over which IP addresses must appear to be considered frequent, this approach, on the other hand, will exclude some legitimate clients from the IP address database. Subsequently, the number of legitimate clients allowed to access the service is reduced. In other words, the effectiveness of defending against spoofed DDoS attacks is not significant since some frequent clients may be prohibited from accessing the protected service.

Recall that in PI [115], each participating router marks some bits (one bit or two bits) in the Identification field of an IP packet according to the router's IP address and the TTL value in the IP header. In this way, an IP packet will arrive at its destination along with a unique identifier representing the path it has traversed. Since the marking is deterministic, packets traversing the same path will share an identical path identifier. With this scheme, the path identifier of a single identified attack packet will provide the victim the ability to filter subsequent attack packets with the same path identifier. However, the motivation of using this approach is unclear. Since the victim is capable of detecting a single attack packet, the reason for not using the same detecting facility to detect subsequent attack packets is not mentioned. Moreover, consider that packets traversed the same last 8 routers before they enter the Autonomous System (AS) at which the victim resides, the victim will not be able to distinguish these packets since their path identifiers are identical (in the case of each router marks two bits in the Identification field). As a result, IP packets of legitimate clients that traverse the same last 8 routers with attack packets will be discarded. Furthermore, consider a special case that the number of participating routers between an attack and the victim is smaller then 8. The attack can partially pollute the attack mark list, which represents the marks of attack packets. It is because, in this case, some bits in the Identification field are under the control of the attacker and remain unmarked throughout the path. This may result in mis-classifying a large portion of legitimate packets as attack ones. Finally, this scheme cannot handle DDoS attacks originated from the AS at which the victim resides, since the marking operation is suppressed on routers in the AS.

To effectively detect and filter spoofed DDoS attacks, we propose a new anti-DDoS scheme, called ANTID. ANTID focuses on the identification of spoofed IP packets and the filtering of attack packets when a DDoS attack occurs. By weeding out spoofed IP packets constituting a dominant share of DDoS attack traffic, DDoS attackers are forced to use real source IP addresses in attack packets. This allows packet filtering mechanisms to discard packets according to their source IP addresses. Moreover, sophisticated resource management schemes can be used in conjunction with the proposed scheme to sustain the quality of protected Internet services.

Our scheme is inspired by hop-count filtering scheme [57] and path identification scheme [115]. In the proposed scheme, each Internet router participating in the defense of DDoS attacks deterministically marks each incoming IP packet such that every IP packet can arrive at its destination along with a unique *path fingerprint* representing the route it has traversed. (In this dissertation, routers that participate in DDoS defense mechanisms are referred to as *participating routers*.) As we shall see shortly in this section, though there are other approaches attempting to create a unique path identifier for each Internet path, they are somewhat weak in defending against sophisticated DDoS attacks. Our approach proposed a new method to create such unique identifier and can resist those sophisticated attacks.

Since a spoofed IP packet is unlikely to have a path fingerprint identical to that of the source IP address being spoofed, a destination host can identify a majority of spoofed IP packets and then discard these packets when it is under spoofed DDoS attacks. From this point of view, establishing the mapping table, which contains the mappings from communicating peers' IP addresses to their corresponding path fingerprints, is essential to the effectiveness of our approach. Theoretically, it seems that the mapping table should contain the mappings of all live IP addresses so that an Internet server under attacks can judge IP packets sent from every possible IP addresses on the Internet. However, from a practice perspective, it is usually unnecessary to do so since the set of IP addresses which frequently visit a normal site usually takes a relatively small portion of all live IP addresses [58, 85]. At the same time, building such a mapping table containing only frequently contacted clients will greatly reduce the storage requirement and lookup time of an Internet server. Thus, in addition to the proposed scheme for identifying/filtering spoofed IP packets with path fingerprints, we give an efficient method that enables an Internet server to construct and update the database in an on-demand fashion. In this way, a mapping of an IP address is created or updated only when the Internet server receives an IP packet from the IP address or when there are changes on the Internet path between the server and the IP address.

There are two execution modes in the proposed scheme, namely *monitor mode* and *filter mode*. By default, the proposed scheme stays in the monitor mode. In this mode, the proposed scheme collects and updates the path fingerprints of clients who want to connect to the protected Internet server. No spoofed packet is discarded in this mode. However, once the rate of spoofed packets received exceeds a pre-defined threshold, the proposed scheme switches to the filter mode. In the filter mode, spoofed packets and IP packets sent from infrequently contacted clients (that is, clients whose IP addresses and correspondent path fingerprints are not yet recorded) are discarded to guarantee service quality to frequent clients.

ANTID has the advantages of strong incremental deployment property, lightweight processing load for marking, decoding and filtering, strong incentive of deployment. It does not require cooperations between ISP networks, and the filtering of spoofed DDoS packets is performed on a per packet basis. ANTID also possesses other useful characteristics that are not present in other schemes [57,115]. First, it can maintain high accuracy (i.e. low false negative ratio and low false positive ratio) even when under a sophisticated DDoS attack (Details will be presented in Section 3.2.). Second, it can differentiate Internet paths in which the last 8 or 16 routers are identical. Third, the proposed scheme works well even when attackers are near the victim (in terms of number of hops) and conventional schemes cannot work well. According to the experiment results, ANTID can identify 99.95% spoofed packets. Experiment results also indicate that the higher an aggregated attack rate is, the sooner the attack can be detected.

Note that ANTID is designed to defend against DDoS attacks which mainly consisted of spoofed packets. In a DRDoS attack, the attack packets sent to the victim server are generally not spoofed and can be handled by conventional schemes. In this case, DRDoS victim will not directly benefit from ANTID. However, since the packets for triggering a DRDoS attack (the packets delivered to the reflectors) are generally spoofed, our approach can detect the spoofed packets, and make it very difficult for attackers to collect a sufficient large number of reflectors. In other words, with a wide deployment of our scheme, the DRDoS attack can also be hampered.

The chapter is organized as follows. Section 3.2 presents a new attack that can circumvent conventional DDoS defense schemes. Section 3.3 presents the details of the proposed path fingerprint scheme. Section 3.4 analyzes the robustness of our approach against attacks. Section 3.5 presents experimental results and this chapter concludes with Section 3.6.

3.2 New Attacking Technique

In this section, we propose a new attacking technique that can used to circumvent hop-count-filtering approach. Specifically, we will illustrate using IP SOURCEROUTE option and ICMP echo-request/echo-reply messages to explore the list of intermediate routers between two remote systems. In this way, an attacker would obtain the hop-count information that is sufficient to dodge hop-count filtering.

We will first give a scenario for obtaining the hop-count information between two remote hosts. Later on in this section, a concrete example will be demonstrated to show the feasibility of this scenario. Fig. 3.1 depicts a simple spoofed attack scenario that an attacker **A** attempts to send spoofed IP packets to a victim **V**, with source IP address **S**. To dodge the hop-count filtering mechanism installed at **V**, the attacker **A** must acquire the number of intermediate routers between **S** and **V**. To achieve this objective, **A** must first obtain the IP address of the default gateway of **S**, i.e. the IP address of **R2**. (Here, it is assumed that an Internet host has the same default gateway for both inbound and outbound traffic. In other words, the ingress and egress routers of an Internet host are identical.) As shown in step (1) of the figure, **A** can easily obtain the IP address of **R2** by using traceroute to obtain the list of routers between **A** and **S**. Next, as shown in step (2), to acquire the number of hops between **S** and **V**, **A** can issue another traceroute command to explore the list of routers between itself and **V**. By setting the IP SOURCEROUTE option on, traceroute packets are forced to traverse **R2**, and **A** can successfully obtain the number of intermediate routers and their IP addresses between **S** and **V**.

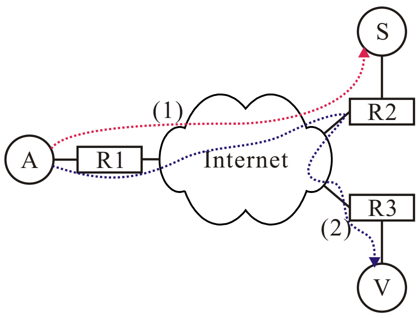


Figure 3.1: A two steps scenario for remotely exploring the number of hops between two end hosts

To illustrate the feasibility of the scenario presented above, an example of exploring the list of routers between two remote sites is demonstrated. The following example is conducted by using the traceroute program in FreeBSD 5.0. In the following example, **A** stands for "140.113.209.21" and **S** stands for "140.112.2.100". Then, **V** is "140.113.216.190". First, the attacker in "140.113.209.21" explores the default gateway of "140.112.2.100" by issuing the command *traceroute 140.112.2.100*. Fig 3.2(a) shows the result of this command, and according to this figure, the IP address of the default gateway of **S** is "140.112.1.13". Afterwards, the attacker issues another traceroute command: *traceroute -g 140.112.1.13 140.113.216.190*, and the correspondent result is shown in Fig. 3.2(b). By comparing the two traceroute results, the attacker can easily find that the number of hops between the two host is 9. Consequently, the attack can successfully dodge the hop-count filtering mechanism by setting an appropriate initial TTL value in the IP packet header.

The aforementioned example shows that hop-count filtering is vulnerable to sophisticated DDoS attacks that can explore the hop-count information between the victim and spoofed IP addresses. Thus, developing a new technique for defending against sophisticated spoofed DDoS attacks is needed.

```
> traceroute -n -q 1 140.112.2.10
traceroute to 140.112.2.100 (140.112.2.100),
64 hops max, 44 byte packets
1 140.113.209.254 0.327 ms
2 140.113.0.210 0.200 ms
3 140.113.0.166 0.228 ms
4 140.113.0.98 0.238 ms
5 192.83.196.111 1.581 ms
6 203.72.43.210 1.537 ms
7 203.72.43.252 1.712 ms
8 140.112.1.29 1.780 ms
9 140.112.1.13 1.669 ms
10 140.112.2.100 1.681 ms
```

(a)

```
> traceroute -n -q 1 -g 140.112.1.13
140.112.216.190
traceroute to 140.113.216.190
(140.113.216.190), 64 hops max, 52 byte
packets
1 140.113.209.254 13.651 ms
2 140.113.0.210 4.727 ms
3 140.113.0.166 3.450 ms
4 140.113.0.98 2.490 ms
5 192.83.175.111 2.144 ms
6 203.72.43.210 2.553 ms
7 203.72.43.252 2.243 ms
8 140.112.1.29 2.385 ms
9 140.112.1.13 3.078 ms
10 140.112.1.14 2.490 ms
11 140.112.1.114 2.889 ms
12 203.72.43.254 3.179 ms
13 203.72.43.209 3.046 ms
14 192.83.196.113 2.898 ms
15 140.113.0.97 2.854 ms
16 140.113.0.165 3.786 ms
17 140.113.0.209 3.105 ms
18 140.113.216.190 3.453 ms
```

(b)

Figure 3.2: (a) An example of determining the default gateway of an IP address being spoofed. (b) An example of enumerating the list of routers between a spoofed source and the victim

3.3 Proposed Path Fingerprint Scheme

In this section, we propose a new spoofed packet filtering anti-DDoS scheme, called ANTID. In the scheme, each IP packet is embedded with a unique *path fingerprint* representing the route an IP packet has traversed, and IP packets with incorrect path fingerprints is considered spoofed. The proposed scheme eliminates some weaknesses of conventional schemes, and is designed specific for defending against spoofed DDoS attacks. It intends to complement, rather than replace existing schemes.

The basic of ANTID is the validation of an IP packet via its source IP address and the path fingerprint embedded in it. In this section, the computation of a path fingerprint is first described, and then the inspection algorithm for identifying spoofed IP packets is presented. Next, an efficient approach for constructing a table that contains the mappings of IP addresses and their path fingerprints is proposed. Finally, the details of detecting a spoofed DDoS attack are shown, and subsequent packet filtering operations are examined.

3.3.1 Path Fingerprinting and Spoofed Packet Inspection

To generate a path fingerprint representing the route an IP packet traversed, it is assumed that each participating router assigns each of its network interface a n-bit random number, and these random numbers are *kept securely*. These numbers should not be disclosed. Additionally, we assume that Internet routers are secure, and it is hard for malicious users to eavesdrop traffic by tapping on network links between Internet core routers.

In ANTID, a *path fingerprint* of an IP packet is composed of two fields, a d-bit *distance* field and a n-bit *path identification* (PID) field, where the former represents the number of intermediate routers traversed, and the latter denotes an identifier derived from the random numbers associated with the traversed network interfaces in the route. The path fingerprint of an IP packet is stored in the IP packet header, and thus it is delivered to the destination host along with the packet. Moreover, we also assume that a *pf-flag* bit in the IP packet header is available for indicating the start of path fingerprinting. Later in this section, we will discuss the allocation of $(1 + d + n)$ bits in the IP packet header fields.

The path fingerprinting procedure is presented as follows. Whenever a participating router receives an IP packet, it first examines the pf-flag field. If it is unset, i.e. 0, the receiving router is then aware of that it is the first participating router the packet encountered in the path. In this case, the receiving router sets the pf-flag bit to 1, sets the distance field to 1 and sets the path identification field to the random number associated with the incoming interface of the packet. On the other hand, if the flag bit is already on, i.e. 1, the receiving router increments the distance field by one, and updates the path identification field with H(PID, N_i), where PID represents the current value of path identification field in the packet, N_i denotes the random number of the incoming interface, and H is a one-way hash function with weak collision resistance. (Note that H is not a secret and each participating router can choose its hash function.) Algorithm 1 shows the pseudo-code for computing path fingerprint on a participating router, and Fig. 3.3 illustrates an example of the path fingerprinting scenario.

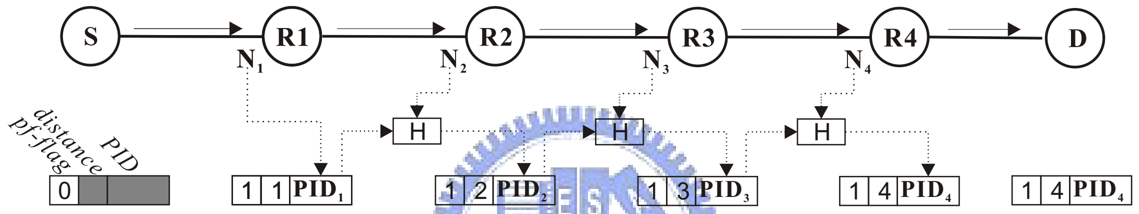


Figure 3.3: An example of the proposed path fingerprinting scheme.

Algorithm 1 Computation of path fingerprints on a participating router

```
1: Let $p$ denote an incoming IP packet.
2: Let $p.pf\text{-}flag$, $p.distance$ and $p.pid$ denote the pf-flag, distance and path identification fields
   in packet $p$, respectively.
3: Let $N_i$ denote the random number associated with the incoming interface of $p$.
4: if $p.pf\text{-}flag$=0 then
5:    $p.pf\text{-}flag \leftarrow 1$
6:    $p.distance \leftarrow 1$
7:    $p.pid \leftarrow N_i$
8: else
9:    $p.distance \leftarrow p.distance + 1$;
10:   $p.pid \leftarrow H(p.pid, N_i)$;
11: end if
```

In the example depicted in Fig. 3.3, a packet traverses from the source **S** to the destination **D** across Internet routers **R1** to **R4**. As aforementioned, we assume that these Internet routers are secure against compromising and it is hard for attackers to eavesdrop packets transmitted via a network link connecting two Internet routers. The first router in the path, **R1**, set both pf-flag and distance filed to 1 and set the initial

PID value to the random number of the incoming interface, i.e. N_1. Afterwards, each router increases the distance field and updates the PID field according to the previous PID value and the random number of the current incoming interface. In this figure, H denotes a hash function.

To allocate space from the IP packet header for storing a path fingerprint, the 16-bit Identification field in the IP header is chosen to be overloaded. Issues related to the overloading of this field has been studied and reported [97]. The 16-bit Identification field is divided into two sub-fields. The first sub-field is 5-bit long and is used to store the value of distance. It is believed that 5 bits is sufficient [14, 105] since most of Internet paths are shorter than 31 hops. To deal with Internet paths with more than 31 hops, a simple solution is to user more bits. However, this will reduce remaining bits for storing path identification, and consequently increase the collision rates. To avoid increasing hash collisions, in our scheme, we choose to stop increment the distance field when its value reaches 31. Though in this case, Internet paths that have more than 31 routers supporting our scheme will have the same distance value, the path identification field can still help distinguish them if their path identifications are different. The remaining 11 bits of the Identification field are used to store path identification. Finally, we propose to use the un-used bit of the FLAG field in IP header to store the value of the pf-flag bit.

In this way, filtering spoofed IP packets will be quite straightforward if the table that contains the mappings of IP addresses and their path fingerprints is present. In this dissertation, the table is referred to as *S2PF* (abbrev. of Source to Path Fingerprint) table. Here, we first assume that the S2PF table is available, and its construction will be discussed later.

It is worthy to note that currently we assume that the mapping from of IP addresses and their path fingerprints is one-to-one mapping. That is, in the S2PF table, each source IP address occupies a S2PF table entry and has its own path fingerprint. It is clear that this approaches would lead to a large S2PF table even if many IP addresses have identical path fingerprints. One possbile solution to this problem is to use many to one mapping, and this technique have been introduced in [57]. That is, IP addresses mapping to identical path fingerprints are collected and expressed using CIDR format. Because hosts in the same subnets generally traverse the same Internet path to a specific destination, their path fingerprint will thus be identical. From this point of view, by aggregating IP addresses with the same path fingerprint allows storing more mappings without incuring storage expansion. For instance, a mapping from an IP address "140.113.216.141" to its path fingerprint can be rewritten as a mapping from "140.113.216.141/8" to the path fingerprint. This helps reduce storage requirement of a S2PF table, and other details are discussed in [57].

It is worthy to mention that our scheme does not have to be deployed on all Internet routers. It is clear that the proposed scheme can be deployed incrementally since each participating router can operate independently. A basic principle for deployment is to ensure that any Internet route will pass through at least one participating router. This makes every IP packets marked by at least one router and thus carry a path fingerprint. Subsequently an end host is capable of inspecting the validity of each incoming packets. From this point of view, the proposed scheme should be first deployed on edge routers or border gateways. This helps minimize the cost of deployment and provides a level of guarantee that each packet will be marked.

Algorithm 2 shows the pseudo-code for identifying spoofed IP packets. In the algorithm, the source IP address and path fingerprint of an IP packet is extracted from the IP packet header first. Next, by using the extracted source IP address, the correspondent path fingerprint can be retrieved from the S2PF table. If the path fingerprint of the given IP address cannot be found in the S2PF table, the algorithm returns *UNKNOWN*. Otherwise, the algorithm compares the two path fingerprints. (One from the IP packet and the other from the S2PF table.) If they are identical, the algorithm returns *LEGITIMATE*, or else, it returns *SPOOFED*.

Notice that the inspecting algorithm only determines whether a given IP packet is spoofed or not. Weeding out spoofed packet is not performed by ANTID alone. This is because the inspecting algorithm may mis-classify legitimate packets as spoofed ones when an out-of-date S2PF table is in use. (A S2PF table will become out-of-date when there are topological changes in the Internet, or a participating router re-assigns random numbers to its network interfaces.) To avoid errors caused by an obsolete S2PF table, spoofed packets will only be discarded after a DDoS attack signal is caught by ANTID.

Algorithm 2 Spoofed Packet Inspection

```
1: Let $src$ denote the source IP address and $pf$ denote the path fingerprint of a given IP packet.
2: Retrieve the path fingerprint,$pf_{s2pf}$ indexed by $src$ in the S2PF table.
3: if No entry indexed by $src$ found then
4:     Return UNKNOWN
5: else if $pf_{s2pf} \neq pf$ then
6:     Return SPOOFED
7: else
8:     Return LEGITIMATE
9: end if
```

3.3.2 The Construction and Update of the S2PF Table

As mentioned previously, there are two execution modes in the proposed scheme, namely *monitor mode* and *filter mode*. In the monitor mode, the S2PF table is constructed, and its entries will be updated if there are changes in the topology of Internet or in the Internet paths due to dynamic routing. Notice that, ANTID does not attempt to build a S2PF table containing all live IP addresses. Instead, the S2PF table should only have entries for IP addresses that ever connected to the destination in the past communications. It is believed that S2PF constructed in this way is sufficient enough because, according to the report [58], the source IP addresses of a given site in normal conditions only take a small set of values. Thus, in ANTID, a new S2PF table entry will be added if the destination host receives IP packets from a new IP address. Additionally, for applications that has a predetermined set of clients, the mapping from legitimate clients' IP addresses and their path fingerprints can be maintained in the S2PF table. Spoof packets with mismatching path fingerprint can be easily detected and then be discarded. This allows controlling the size of the S2PF table at a manageable level, and, at the same time, reducing the time for searching.

There are different ways to learn the mapping of an IP address and its path fingerprints from communications. One naive approach is to learn the mappings simply from received IP packets. Whenever an IP packet with a new source IP address arrives, a new entry is inserted into the S2PF table. Similarly, the arrival of an IP packet that carries a new path fingerprint different from the one already stored in the S2PF table would result in an update on the correspondent S2PF entry. In this way, constructing a S2PF table is quite straightforward, however, it is clear that this naive approach will not work for identifying spoofed packets since no validation process is involved. A more complicated method is to learn the mappings from established TCP connections. Since it is very difficult to spoofed source IP addresses in TCP connections, the validity of mappings learned in this way is ensured. However, this method is not appropriate for Internet servers that do not run TCP-based services. Herein, we suggest a simple and generic method to obtain the mappings in an efficient and robust manner. That is, the path fingerprint of a specific source IP address is explicitly explored by the use of ICMP echo-request messages. Before an entry of a new source IP address can be inserted into the S2PF table or an update can be made, the destination host sends an ICMP echo-request message to the source IP address. Then, the path fingerprint in the returned ICMP echo-reply message is treated as the most up-to-date path fingerprint of that source IP address. It is worthy to note here that traceroute packets might be blocked for security reasons. Thus, an alternative way is to use the ICMP port unreachable message and the TCP RST message. Specifically, the destination host can send a UDP packet to that specific source IP address and the source port number and destination port number can be set arbitrarily. In this case, it is very likely that no process is serving on that destination port, and thus an ICMP port unreachable message will be sent back to the server. In that ICMP packet, the victim server can learn the path fingerprint between itself and the specific host. Similarly, sending a TCP packet with randomly created destination port number can be used to accomplish the same objective. The corresponding TCP RST packet will carry the path fingerprint. No matter which approach is taken, The S2PF table, constructed in this way, will contain only correct mappings of legitimate clients.

There are two main reasons for invoking exploration of the path fingerprint of a specific IP address. The first refers to the arrival of an IP packet with a new IP address. The second directs to the necessity of updating a S2PF table entry. Consider the first case when the packet from a new client arrives. In this case, the spoofed packet detection algorithm presented in Algorithm 2 returns *UNKNOWN*, and an exploration process will be invoked at probability q, where $0 \leq q < 1$. In this way, we can avoid overloading the Internet and can avoid building a S2PF table containing a large portion of infrequently contacted clients. As to the second case, although the majority of Internet paths are expected to be stable and remain unchanged for a long period of time [57,83]; occasionally it is still necessary to update the S2PF table when the routing changes. This update function is important to maintain an up-to-date S2PF table. In the proposed scheme, upon receipt of an IP packet that traversed a new Internet path (assuming that the S2PF table has an entry for the IP address of this packet), the spoofed detection algorithm will classify this packet as spoofed. Then, in this case, an exploration process will be invoked at probability r, where $0 \leq r < 1$. Both q and r are used to prevent our scheme from excessively exploring path fingerprints by ICMP echo-request/echo-reply messages, and at the same time, we preserve the ability to insert and update entries in the S2PF table.

Since a S2PF table cannot accommodate the mappings of all possible IP addresses (there can be at most 2^{32} entries), replacing an old S2PF table entry with a new one is also an important issue that needs to be addressed. Whenever a replacement is needed, we currently recommend that several cache replacement techniques, such as *Least Frequently Used* (LFU), *Least Recently Used* (LRU) and *Most Frequently Used* (MFU), can be used. However, in this dissertation, we do not make definitive claim that which replacement technique is the best for the S2PF table entry replacement, and determining the best replacement policy warrants further research.

Finally, in ANTID, the number of spoofed packets received is used as a criterion to determine the onset of a spoofed DDoS attack. Thus, in the monitor mode, whenever an exploration process is invoked owing to receipt of a new IP address, the returned path fingerprint is compared against with the path fingerprint stored in the IP packet. If they are not identical, a counter *spoofing-cnt*, which records the number of spoofed packet received in one unit of time, will be increased by one. Similarly, whenever the inspecting algorithm returns *SPOOFED*, the spoofing-cnt is also incremented by one unless the exploration process returns an path fingerprint identical to the one in the IP packet. Algorithm 3 shows the pseudo-code for the construction and update of the S2PF table in the monitor mode.

3.3.3 State Transitions and Spoofed Packet Filtering

As mentioned, the number of spoofed packets received is used as a criterion for transition between the monitor mode and the filter mode. In the proposed scheme, the time is divided into a set of uniform time intervals. At the beginning of each time interval, the spoofing-cnt, which records the number of spoofed packet in the previous time interval, is examined. If the value of this counter exceeds a threshold T_1, ANTID switches to the filter mode. In the filter mode, spoofed packets and packets with source IP addresses absent

Algorithm 3 The construction and update of the S2PF table

```
1: Let $p$ denote the incoming packet.
2: Let $p.src$ and $p.pf$ denote the source IP address and the path fingerprint stored in the packet header of $p$.
3: Status ← SpoofInspection(p).
4: if Status = UNKNOWN then
5:   Let $x$ be a random number from $[0, 1)$.
6:   if $x < q$ then
7:     Invoke the path fingerprint exploration process.
8:     Insert a new table entry of $p.src$ into S2PF table.
9:     Let $p.pf_{new}$ denote the newly acquired path fingerprint of $p.src$
10:    if $p.pf_{new} \neq p.pf$ then
11:      Classify packet $p$ as spoofed
12:      $spoofing\text{-}cnt \leftarrow spoofing\text{-}cnt + 1$
13:    end if
14:  end if
15: else if Status = SPOOFED then
16:  Let $x$ be a random number from $[0, 1)$
17:  if $x < r$ then
18:    Invoke the path fingerprint exploration process.
19:    Update the entry of $p.src$ with newly acquired path fingerprint.
20:    Let $p.pf_{new}$ denote the newly acquired path fingerprint of $p.src$
21:    if $p.pf_{new} \neq p.pf$ then
22:      $spoofing\text{-}cnt \leftarrow spoofing\text{-}cnt + 1$
23:    end if
24:  else
25:    $spoofing\text{-}cnt \leftarrow spoofing\text{-}cnt + 1$
26:  end if
27: end if
```

Algorithm 4 The transition between monitor and filter modes

```
1: In the begin of each time interval
2: if current execution mode = monitor then
3:   if $spoofing\text{-}cnt > T_1$ then
4:     switch to filter mode
5:   end if
6: else
7:   if $spoofing\text{-}cnt < T_2$ then
8:     switch to monitor mode
9:   end if
10: end if
11: $spoofing\text{-}cnt \leftarrow 0$
```

Algorithm 5 Spoofed Packet Filtering

```
1: Let $p$ denote an incoming IP packet.
2: Status=SpoofedInspection(p)
3: if Status = SPOOFED or UNKNOWN then
4:   Drop the packet p
5:   $spoofing\text{-}cnt \leftarrow spoofing\text{-}cnt + 1$
6: end if
```

in the S2PF table are discarded. The proposed scheme switches from the filter mode to monitor mode when IP spoofing ceases. This is achieved by examining the spoofing-cnt. If the value of this counter is smaller than another threshold T_2, ANTID switches back to the monitor mode. In this dissertation, we provide only a general guideline commonly used in threshold schemes for setting the two thresholds. The basic principle is to let $T_1 > T_2$. It is clear that this can prevent ANTID from alternating between the two execution modes. Another important concern is that T_1 should be set appropriately such that the victim server will not falsely switch into the filter mode. Switching to the filter mode too easily may lead to another form of DoS attack because, in the filter mode, the victim server only serves previously validated clients.

As to problem of setting the specific values of these parameters, such as q, r, T_1 and T_2, we recommend that they should be configurable by administrators of the Internet servers. These parameters are highly application-dependent. That is, it is largely relied on administrators to determine their own best trade-off between the performance and security of the protected sites. Herein, we only briefly enumerate some factors related to the setting of these parameters. First, q is related to definition of a "frequent" visitor. q can be set to 1/10 if a user is considered a frequent users when the users visits the protected site for more than 10 times. Second, r highly depends on the dynamics of Internet topology. Though Internet paths were found to be persistent for days, the Internet topology is assumed to change dynamically. Further investigation on the dynamics of the Internet is needed to provide hints for setting r appropriately. As for T_1 and T_2, their values highly depend on the number of spoofed packets the protected site can tolerate in each time interval. They can be determined by the maximum number of packets arriving in each time period in victim's network link or the maximum number of service request the server can accept. For instance, consider a Internet server whose maximum packet arrival rate is about 5000 packets per second. Then, T_1 can be set as 2500 (about the 1/2 of the average number of packets), and T_2 can be set as 500 (about the 1/10 of the average number of packets). Other criteria can be used to determine T_1 and T_2. Or, if the server can process 10000 packets per second, then T_1 can be set as 5000 and T_2 can be set as 1000. We believe that such settings would be useful in detecting large scale spoofed DDoS attacks that attempt to flood the protect site with spoofed packets.

Algorithm 4 shows the pseudo-code of state transition, and Algorithm 5 presents the pseudo-code of spoofed packet filtering in the filter mode.

3.4 Robustness against Circumvention

In this section, we present possible approaches that a DDoS attacker may take to circumvent the proposed DDoS defense mechanism and show that our scheme is robust against these attacks. The key for an attacker to circumvent spoofed packet detection is the ability to generate attack packets in accordance to the constraint that they can finally arrive at the victim along with path fingerprints consistent with the spoofed IP addresses. In the following, we examine two types of approaches to achieve this objective.

- **Simple Attack:**
  The simplest approach is to set random initial values in both the path fingerprint field and the source IP address field. Notice that an attacker *cannot* seed the fingerprint field of attack packets in such a way that the fingerprint of the attack packets will arrive at the victim server along with a correct path fingerprint. This is because the value in the path fingerprint field will be changed securely. Without knowing all the random numbers associated with the traversed links, the attacker has no knowledge of the correct seed. In other words, since the random numbers associated with network links are kept securely, it is very difficult for an attacker to control path fingerprint received by the victim. Thus, the best an attacker can do is to set random seed in the fingerprint field and the source IP field. However, this approach is infeasible since it is very unlikely that the randomly spoofed source IP address will exist in the S2PF table at the destination host or that these attack packets can arrive at the destination along with a correct path fingerprint. The probability for a match is $1/2^{(d+n)}$ (in our scheme, $(d+n)$=16). Next, consider a more sophisticated case that an attacker can carefully select a spoofed IP address and can set an appropriate value in the distance field (setting an appropriate initial value in the distance field can be achieved by using the technique presented in Section 3.2). In this case, only very few attack packets can pass the spoofed packet detection. It is because that the best an attacker can do is to fill the path identification field with a random value and then only the fraction $1/2^n$ (in our scheme, n=11) of attack packets can arrive at the destination along with a correct

path identification value. In short, such a simple attack is not useful to dodge the proposed scheme.

- **Detour attack:**
  As we have shown in Section 3.2, an attacker can determine the default gateway of a spoofed IP address. Thus it is reasonable to assume that an attacker can force attack packets to traverse the default gateway of the spoofed IP address by using IP SOURCEROUTE option. In this way, the postfix of the attack path will be identical to the path from the spoofed source to the victim. This type of attack is referred to as *detour attack*. The success of a detour attack relies on the following mandatory conditions: there must not exist any participating router in the path from the attacker to the spoofed IP address (including the default gateway of the spoofed source). If this condition holds, an attacker can successfully conduct a spoofed DDoS attack by using the detour technique presented here. In this case, the victim cannot identify spoofed attack packets since the path fingerprints of these packets are correct. Although this type of attack allows an attacker to dodge path fingerprint filtering, finding an appropriate spoofed source is very difficult if the participating routers are widely distributed over the entire Internet. Moreover, the victim can easily stop attack packets of a detour attack by filtering IP packets with IP SOURCEROUTE option set.

According to the above analysis, we can find that both the simple attack and the detour attack are ineffective. Furthermore, from a probability point of view, in the simple attack, attack packets can bypass the spoofed packet detection at probability of $1/2^{11}$. (Later on we will confirm this by experiments.) In summary, the proposed scheme is robust against circumvention.

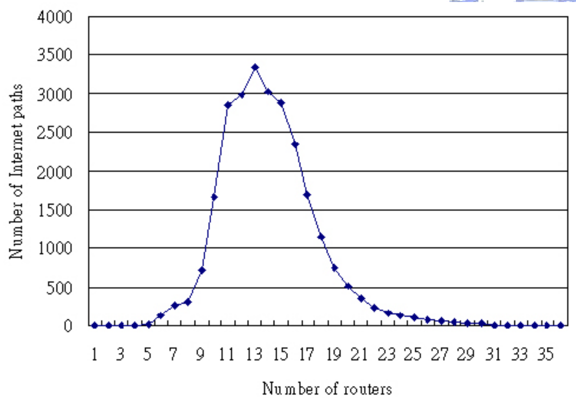
3.5 Evaluation

In this section, we evaluate the accuracy of the proposed scheme in the identification and filter of spoofed packets under DDoS attacks. We simulate the the aggregate of DDoS attack traffic at different attack rates and then present the performance metrics that we measure.

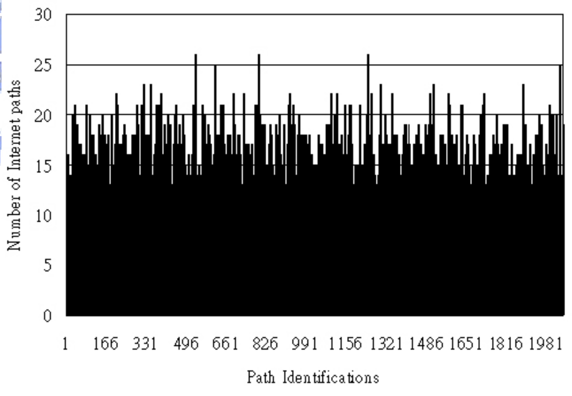
3.5.1 Internet Data Sets

To simulate the Internet topology, the Internet map [30] is used as our Internet topological data. The Internet map contains Internet paths (each represented as a list of routers), from a specific host to most of nets on the Internet. In our experiments, the host which originates these traceroute-style path probes is viewed as the victim of a DDoS attack, and attackers and legitimate clients are randomly selected from those hosts at the end of each traceroute path.

Figure 3.4(a) depicts the distribution of number of intermediate routers on each completed Internet paths. There are in total 24772 Internet paths. (We exclude incomplete traceroute probes from our experiments.) As the figure shows, only a few Internet paths consist of more than 32 intermediate routers, and the most popular path length is 16. Figure 3.4(b) shows the distribution of the path identifications of these Internet paths. Theoretically, since the numbers associated with network interfaces are randomly assigned, the path identifications of Internet paths should also be random, and Fig. 3.4(b) confirms this theoretical inference.



(a)



(b)

Figure 3.4: (a) The distribution of number of intermediate routers. (b) The distribution of the value of path identifications

3.5.2 Experimental Design and Performance Metrics

In the experiments, 500 end hosts are randomly selected from the Internet map to act as frequently contacted clients of an Internet server. Then, a S2PF table which contains the "source to path fingerprint" mappings of the 500 clients is constructed. Moreover, we set q to 1/10, r to 1/100 and T_1 to 2500. Notice that, in our experiments, we do not attempt to find a best suite of values of these configurable parameters for a specific network environment. (As indicated in Section 3.3, the setting of these parameters heavily depends on both specific characteristics of deployed networks and the trade-off between security and performance. Thus, we do not focus on this issue in this dissertation.) Instead, other behaviors, such as the growth rate of the S2PF table and the false negative ratio at the monitor mode and the filter mode, are explored under various attack rates.

First we measure the false negative ratio of the proposed scheme before and after it switches from monitor mode to filter mode. Herein, the false negative ratio refers to the ratio of undetected spoofed packets. In this experiment, we simulate the aggregate of attack traffic that have 5000 attack packets in each attack round. (Note that the growth of the number of attackers can only increase the aggregation of attack traffic, but cannot increase the probability of passing the proposed filtering mechanism. Thus, we use the aggregation of attack traffic to test the proposed scheme.) A round of attack, in fact, stands for a period of time. In this dissertation, we do not impose restrictions on setting the length of an attack round. Instead, we are interested in the number of rounds and the rate required to detect the presence of a DDoS attack. Source IP addresses of these attack packets are randomly selected from the Internet map with a constraint that these addresses are disjoint with legitimate clients. Moreover, by the same technique presented in Section 3.2, we let these attack packets carry appropriate distance values such that they can arrive at the victim along with consistent distance values with the spoofed IP addresses. As shown in Fig. 3.5, the false negative rate shows a tendency to decrease as the number of rounds increases, and finally at round 3156, the proposed scheme switches to filter mode. Thus, after round 3156, the false negative ratio steeply drop to around 1/2048. The decrease of false negative ratio at the monitor mode is caused by the growth of the size of the S2PF table. Since the victim will add a new S2PF table entry at probability 1/100, about (1/100)$*$ (number of spoofed IP addresses not in the S2PF table) new entries will be added to the S2PF table after each attack round. At round 3156, there are in total 11937 entries (about the half of number of Internet paths in Internet map) in the S2PF table.

As to false negatives, in our approach, legitimate packets will not be classified as spoofed ones in the monitor mode. They will be classified as UNKNOWN packets, or

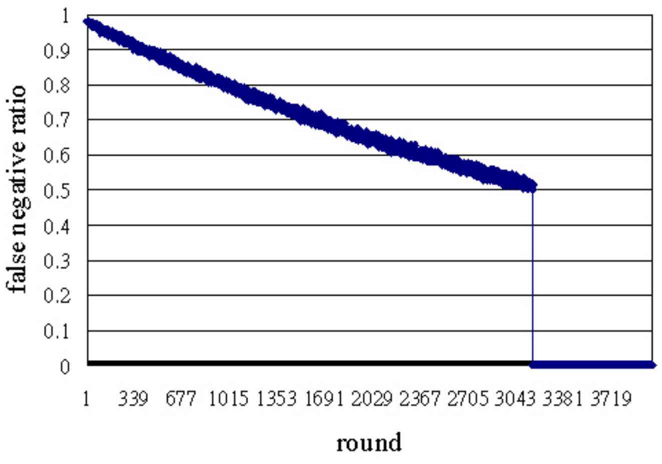
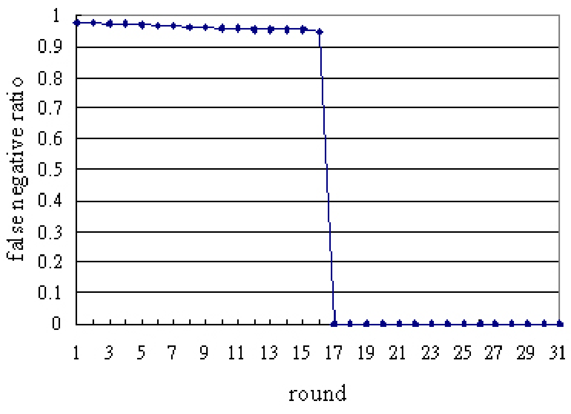


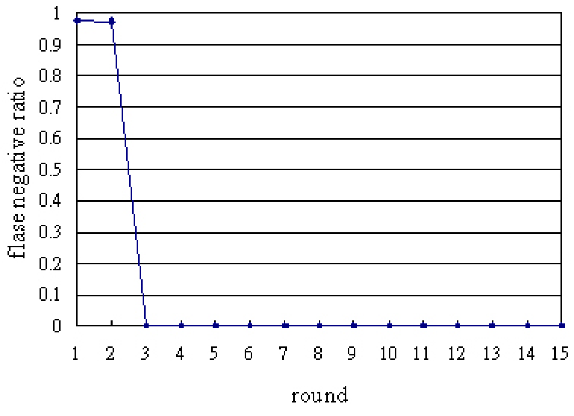
Figure 3.5: The false negative ratio under the attack rate of 5000 packets per round.

their path fingerprints will be inserted into the S2PF table after the fingerprint exploration process. While in the filter mode, our scheme indeed will discard some legitimate packet originated from unfrequent clients whose path fingerprints are not in the S2PF table. These packets are still classified as UNKNOWN rather than SPOOFED. Possible cases of misclassifying legitimate packets as spoofed ones are mostly caused by changes in the Internet topology when our scheme is in the filtering mode. In filtering mode, our scheme stop updating path fingerprints for SPOOFED packets. Such false positives highly depend on the dynamics of the Internet, and therefore are not discussed in the context of this dissertation.

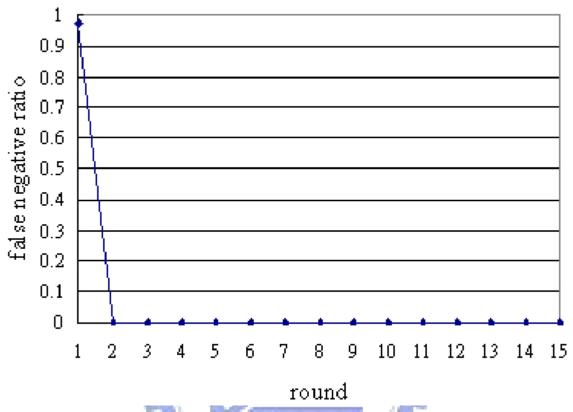
According to the statistics on the measured DDoS attacks, attack rates range from 500 to 600,000 attack packets per second [34,78]. In the following, we will present experiments on how many packets are needed for our scheme to detect the presence of a DDoS attack, and please notice that the sensitivity presented here highly depends on the settings of configuration variables. Fig. 3.6 shows the number of rounds required to detect the presence of a spoofed DDoS attack under three attack rates: 50000, 100000 and 150000 attack packets per round. This figure shows that the number of rounds needed to detect a spoofed DDoS attack decreases as the attack rate increases. Next, we conduct the same experiments at lower attack rates. In these experiments, we send 5000, 10000, 15000, . . . , 50000 attack packets to the victim at each round. We observed the growth of the S2PF table and the number of rounds needed to detected attacks. The experimental result is shown in Table 3.1. According to the table, we can find that the higher the attack rate is, the fewer rounds and the fewer entries in the S2PF table is required to detect the presence of an attack.



(a) attack rate=50000



(b) attack rate=100000



(c) attack rate=150000

Figure 3.6: The false negative ratio under the attack rates of 50000, 100000 and 150000 attack packets per round

Table 3.1: At different attack rates, the number of rounds and the number of table entries required to detect the attack.

Attack rate	5000	10000	15000	20000	25000	30000	35000	40000	45000	50000
Table size	11937	5928	3983	2990	2462	2037	1750	1517	1378	1243
No. of rounds	3156	621	259	136	88	52	37	27	20	16

In summary, the experimental results show that a higher attack rate will result in a smaller number of rounds required to detected an ongoing spoofed DDoS attack. After the attack is detected, only a very small fraction (around 1/2048) of attack packets can pass the spoofed packet detection mechanism. This shows that our approach can effectively detect the presence of an attack and subsequently can weed out these attack packets.

3.6 Summary

In this chapter, we presented an anti-DDoS scheme, ANTID, for defending against spoofed DDoS traffic. ANTID intends to complement, rather than replace existing schemes. For instance, the proposed scheme helps discard spoofed packets before ingress filters are installed on all edge routers. Furthermore, by weeding out a majority of spoofed attack packets, our approach allows some resource management systems, that share resource fair among many participants, to work better.

In our approach, each IP packet is embedded with a unique path fingerprint that represents the Internet path it has traversed. By learning path fingerprints from past traffic, the victim can efficiently establish the S2PF table which contains the mappings of source IP addresses and correspondent path fingerprints of frequently contacted clients. A spoofed packet can be easily identified by consulting the S2PF table since it is very unlikely that a spoofed packet can have a path fingerprint identical to that of the spoofed IP address. Thus, by identifying and filtering spoofed packets, a spoofed DDoS attack can be identified and prevented. This makes the proposed scheme an effective and efficient approach for defending against spoofed DDoS attacks.

ANTID runs in two execution modes, the monitor mode and the filter mode. We simulate DDoS attacks with variable attack rates to evaluate the performance of our approach. Experiment results showed that ANTID can provide protection against spoofed DDoS attack. Only around 1/2048 attack packets can pass the spoofed packet detection and filter mechanism when our scheme stays in filter mode. The experiment results also show that the time required to detect an attack depends on the attack rate of aggregated attack traffic. The higher the attack rate is, the short the time for detecting will be. Finally, our approach possesses several important properties, such as strong incremental deployment and lightweight for marking, decoding and filtering. No cooperation among ISP networks is needed. More importantly, it is robust against sophisticated DDoS attacks, and it is resistant to the deception by nearby attackers. These properties make the proposed scheme a general and robust approach that is feasible to be deployed in the Internet. There are several issues that require further investigations. For instance, a systematic way for configuring parameters, q, r, T_1 and T_2, for a specific network environment is required. And, an efficient approach for maintaining the S2PF table is needed. Finally, the best strategy for deploying participating routers in the Internet needs to be designed and investigated.

Chapter 4

Edge-Based Record Route: A Novel Approach for Single Packet Traceback

The detection of network attacks has drawn considerable attention in recent years. In particular, locating the origin of an attack flow is highlighted because it makes possible preventing or limiting an attack at the source. Several schemes have been presented in the literature that address the traceback of DoS/DDoS attacks. However, tracing an attack with a single IP packet remains unsolved and needs further investigation. In this chapter we present a space-efficient route representation, namely *edge-based route representation*. Based on edge-based route representation, we propose a general purpose single packet traceback scheme which allows a route to be encoded using a few bits. The proposed scheme is backward compatible and can be deployed incrementally. In the scheme, the entire route of a packet is divided into a set of partial paths which are recorded and delivered to the destination host along with the packet itself or by ICMP packets. Although using a large of number of ICMP packets may seemingly pose substantial overhead to routers and networks, our experimental results show that, in practice, this overhead is relatively low. This is because most IP packets will arrive at the destination hosts along with a complete and ordered list of partial paths stored in the IP option field. In this case, very few ICMP packet needs to be created and consequently the overhead is small. Analytic analysis shows that the accuracy of the proposed scheme is high. The probability of identifing the correct attack source reaches 99.6% even if the attacker attempts to deceive our scheme from locating an incorrect source. the accuracy of identifying the attack source This indicates the proposed scheme is effective, accurate, and feasible for the single packet IP traceback.

4.1 Preliminaries

Though detection and access control systems, such as intrusion detection system (IDS) and firewall, are now widely deployed, they can only provide passive protection rather than active countermeasures against network attacks, such as hampering attackers at the attack sources. To defend against DDoS attack in an automatic manner, the first step may be to locate the origin of the attack flow. This allows imposing legal sanctions and consequently protecting other computer systems from being attacked by the same attack source. However, due to the anonymous nature of the IP protocol, it is extremely difficult to determine the origin of an attack flow. To address this issue, several approaches have been proposed in the literature [9,12,38,53,86,96–98,100–102], and these work have been reviewed in Chapter 2. Most of these works focus on tracing the origin of a DoS/DDoS attack, where a large number of network packets are involved. On the other hand, locating the sources of other types of attacks with a small number of packets is seldom addressed. However, in reality some network attacks can disable a network service or crash a remote computer system with only a single IP packet, such as teardrop, land [17], naptha [27], and ping-of-death. To hamper these attacks, the origin of a single IP packet must be identified.

Techniques for tracing DoS/DDoS attacks are difficult to adapt to tracing attacks launched with only a few packets. Current DoS/DDoS traceback approaches heavily rely on a fundamental and unique characteristic of DoS/DDoS attacks, that is, the large volume of attack traffic. Probabilistic packet marking (PPM) [53, 86, 97, 98, 102] and itrace [12] both employs probabilistic methods to locate the origin of an attack flow with a sufficient large number of packets. In both approaches, each packet records a partial path information. After collecting enough packets belonging to an attack flow, a victim can reconstruct the attack path. An algebraic approach [38] addressed the traceback of a DoS/DDoS attack by reframing the traceback problem as a polynomial reconstruction problem. A common requirement of these schemes is that they all require collecting a large number of attack packets before they can reconstruct the attack paths. This requirement inherently makes these schemes infeasible for tracing the source of a single packet.

There are two schemes currently proposed for the traceback of attack flows with a few attck packets. Deterministic packet marking scheme proposed by Blenky [9] employs packet marking technique. Instead of marking packets with a probabilistic method, an IP packet is marked deterministically when entering the network. In other words, each edge router is responsible for assuring the source of IP packets by marking its own IP address in IP packets. However, the requirement of supports from all edge routers poses a great challenge to the deployment of this scheme in large networks, such as the Internet. This scheme also cannot be deployed incrementally, and as a result its effectiveness is somewhat limited. Moreover, they still cannot deal with DDoS attacks in which each attack flow contains only one attack packets.

Hash-based IP traceback proposed by *Snoeren et al.* [96, 100, 101] is a single-packet traceback approach. It employs an auditing technique implemented by computing and storing a digest value of each IP packet in routers. A digest value actually consists of a set of hash values. Each hash value is generated by applying a distinct hash function to selected IP packet header fields and a part of IP payload. By storing digest values in a data structure called *bloom filter* [13], the storage requirement of this approach can be significantly reduced. However, this approach suffers from a fundamental shortcoming. It only supports the traceback of an attack occurring in the recent past. The traceback process can have the chance to complete only in a very limited window of time. In other words, traceback process must be performed in a timely manner; this process will fail if a router in the attack path flushes its bloom filter. Moreover, since false positive increases as the capability of a bloom filter gets higher, it is also hard to ensure the correctness of the candidate attack path returned by hash-based IP traceback. As a whole, existing approaches are considered inadequate to trace single IP packet in large networks due to their infeasible high deployment cost, very limited time constraint, or difficulty in assuring the correctness of identified attack paths. From these reasons, a new solution to the single packet IP traceback is desirable.

In this chapter, a scheme for tracing the origin of a single IP packet is proposed. We first present a novel route representation, namely *edge-based route representation*, which allows a route to be recorded with a small number of bits. With such a route representation, it is possible to store the entire route information along with each IP packet. Subsequently, we detail an implementation of this mechanism. In the implementation of edge-based record route, path information is first recorded in the overloaded IP checksum field. Whenever the checksum filed is fulfilled, currently recorded path information is moved to the IP option field, if applicable. Otherwise, it is delivered to the destination host via ICMP packets. In this way, the route information of an IP packet can be delivered to the destination host.

The overhead of generating a large number of ICMP packets may be seemingly high and may overload routers and networks. However, analytic analysis shows that the number of such ICMP packets is small and the overhead is insubstantial. In most cases, IP packets will arrive at the destination host with a complete and ordered path information. In this case, no ICMP packet is created. In a very rare case that an IP packet traverses over a large number of hops, an ICMP packet is generated. Therefore edge-based record route does not incur substantial overhead for tracing the source of a single IP packet. Another characteristic of the proposed scheme is that it can identify the origin accurately. According to our analysis, the accuracy reaches 99.6%.

The contribution of this chapter is that it proposes an effective way to account for network attacks. It incurs little overhead and does not interfere with existing network infrastructure. The traceback process can be performed following an attack and need not be executed in a very limited window of time. The proposed scheme runs efficiently and can scale up easily in large networks, such as the Internet.

The rest of this chapter examines edge-based record route in details. Section 4.2 presents a general single packet IP traceback algorithm which is based on the proposed edge-based route representation. In Section 4.3, the proposed scheme is given, which allows the route of each IP packet to be traced in a way that can be compatible with existing network infrastructure. In Section 4.4, we present analytic analysis and experimental results for evaluating overhead and accuracy of the proposed scheme. Finally, we summarize and conclude our findings in Section 4.5.

4.2 Edge-Based Record Route

A naive solution for the single packet traceback problem is to have each router append its IP address at the end of a network packet as the packet travels through the network. In this way, every packet arrives at its destination host along with a complete and ordered list of router IP addresses it traversed. Although this approach is conceptually robust and easy to implement, it is infeasible in current network environment. The first practical concern is the high overhead of appending IP addresses in the flight. This may reduce the throughput of routers. Another fundamental shortcoming is the difficulty of assuring that there is always sufficient unused space for the complete list of router IP addresses. Thus, appending IP addresses at the end of a packet may result in unnecessary packet fragmentation when the packet size exceeds MTU [77]. Moreover, in order to avoid being traced, the attacker can always fulfill unused space with false IP addresses to mislead traceback process. Although it is believed that the overhead incurred by appending IP addresses may be overcome with hardware support, the rest of problems make the naive approach become inadequate for single packet IP traceback.

To cope with the single packet traceback problem, we propose a new scheme based on a similar idea which is used in the naive approach, where the entire route of a packet is recorded. In our approach, the two fundamental shortcomings of the naive approach are resolved. First, we propose an effective way that can record a packet's route with a small number of bits. In this way, we can greatly alleviate the overhead and fragmentation of conventional schemes that record IP addresses it traversed. Second, the proposed approach can filter out forged path information which is filled by an attacker.

In this section, the basic idea of our scheme for reducing the space for router recording is first introduced. Then, based on the idea, encoding schemes that convert the space-efficient route representation into binary bit strings are presented.

4.2.1 Edge-Based Route Representation

In this chapter, a network environment is modeled as an undirected linear graph $G = (V, E)$, where the set of vertices V corresponds to routers and end hosts. in the network, and the set of edges E corresponds to physical links between vertices. In the network model, $V = R \cup H$, where R is the set of routers, and the element of H is a set of end hosts attaching to the same physical network interface of a router. An undirected edge (u, v) consists of two directed links $< u, v >$ and $< v, u >$, where in each link the former represents the starting vertex, and the latter represents the terminal vertex. Fig. 4.1 shows an example that a physical network environment is mapped to an abstract network model.

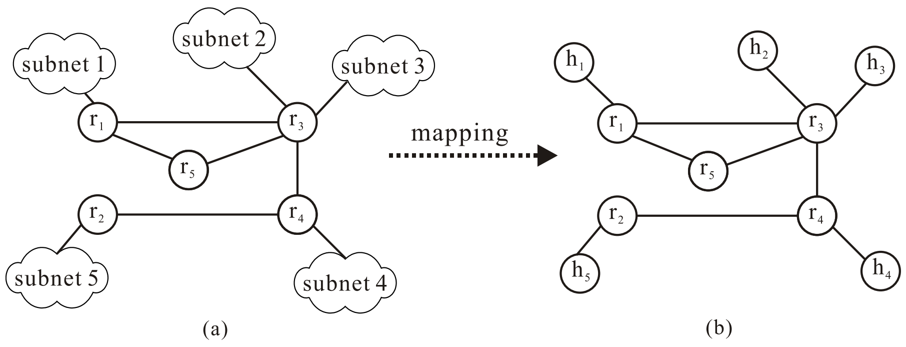


Figure 4.1: Mapping a sample network environment to a network model. (a) shows a sample network with 5 routers and 5 sub-networks. (b) shows the correspondent abstract network model.

Consider a router $r_i \in R$. Let $ip(r_i)$ denote the IP address of r_i, $deg(r_i)$ denote its degree which is the number of links associated with (r_i), and $L(r_i)$ be the set of directed links which is incident into r_i. Each link l_a in $L(r_i)$ is associated with a *identifier* $id(l_a)$ ranging from 0 to $(deg(r_i) - 1)$. For any pair of elements l_a, l_b in $L(r_i)$, $id(l_a) \neq id(l_b)$ and $0 \leq id(l_a), id(l_b) \leq (deg(r_i) - 1)$. Note that identifier $id(l_a)$ is not a global identifier. It is only a local identifier to uniquely identify a local link directly attaching to a router. In this way, size the identifier can be much smaller than IP address, and space to store the identifier in a network packet can be greatly reduced. We will show later on that even without global identifier, the route can be obtained from a set of local identifier. To better understand the edge-based record route, we would first assume that topology of the network is publicly available to all end hosts in the network. Later on, in the proposed scheme, this assumption is relaxed.

Recall that single packet traceback is to determine the sequence of routers (r_{i_1}, r_{i_2}, $\ldots$, r_{i_k}) a given IP packet has traversed from a source host to a destination host. To achieve this objective, as mentioned previously, one can simply have each router append its IP address at the end of the packet. This approach requires space proportional to the number of routers on the path. For instance, consider a path containing k routers, each having a 32-bit IPv4 address, then $k \times 32$ bits in total are needed to store the IP addresses. Herein, we propose a representation of a route, namely *edge-based route representation*, which can significantly reduce the number of bits representing a route. Its definition is given as follows.

Definition 1 (Edge-Based Route Representation) *Let $(l_1, l_2, \dots, l_k)$ denote the sequence of directed links that an IP packet traversed from a source host to a destination host, where each directed link in the sequence is incident into a router. Let r_k be the router l_k is incident into. Then, an edge-based representation $((id(l_1), id(l_2), \dots, id(l_k)), ip(r_k))$ of this route consists of two basic components $(id(l_1), id(l_2), \dots, id(l_k))$ and $ip(r_k)$, where the former represents a sequence of link identifiers, and the latter denotes the IP address of the last router in the path. For simplicity, we relate the route of a packet p to one of its attributes which is denoted as p.path. Then, the two components of an edge-based route representation are represented as p.path.links and p.path.end, respectively. In this way, p.path.links represents the sequence of link identifiers the packet traversed and p.path.end denotes the IP address of the last router which p visited in the path. It is clear that p.path.end can uniquely identify a path.*

The edge-based route representation is constructed by having each router to append the identifier of the incoming link to *p.path.links* and to update the *p.path.end* with its own IP address upon receipt of a packet p. In this way, an IP packet will arrive at the destination host along with an edge-based representation of the route it traversed. This type of record router is referred to as *Edge-Based Record Route (EBRR)* and a router supporting this operation is called an *EBRR-enabled router.*

Consider an end host receives a packet with the edge-based route representation $((id(l_1), id(l_2), \dots, id(l_k)), ip(r_k))$. Let $routers(p)$ be the sequence of routers, which are identified to be in the route of the packet p. $routers(p)$ can be represented as a sequence r_i, $1 \leq i \leq k$. To trace back the route of a packet, initially let $routers(p)$ be an empty sequence, that is, $routers(p) = \phi$. The destination host first inserts *p.path.end* into $routers(p)$, that is, let $p.path.end = ip(r_k)$, and $routers(p) = (ip(r_k))$. According to $id(l_k)$ and r_k, the router, r_{k-1}, in the upstream of r_k can be determined. Consequently, $routers(p) = (ip(r_{k-1}), ip(r_k))$. Repeatedly in this reverse fashion, the sequence of routers can be uniquely determined.

To illustrate the edge-based record route, we will use Figure 2 as an example. Fig. 4.2 shows a sample route of a packet p. Let the sequence of routers in the path be r_1, r_5, r_3, r_4, and r_2. Here we assume that packet p departs from the source with $p.path.links = \phi$ and $p.path.end = 0$. Upon arriving at r_1, $p.path.links = (1)$ and $p.path.end = ip(r_1)$. Subsequently, $p.path.links$ becomes $(1, 2)$ and $p.path.end = ip(r_5)$ when it arrives r_5. This process is repeated until p arrives at its destination host and the edge-based representation of the route of p becomes $((1, 2, 4, 2, 1), ip(r_2))$.

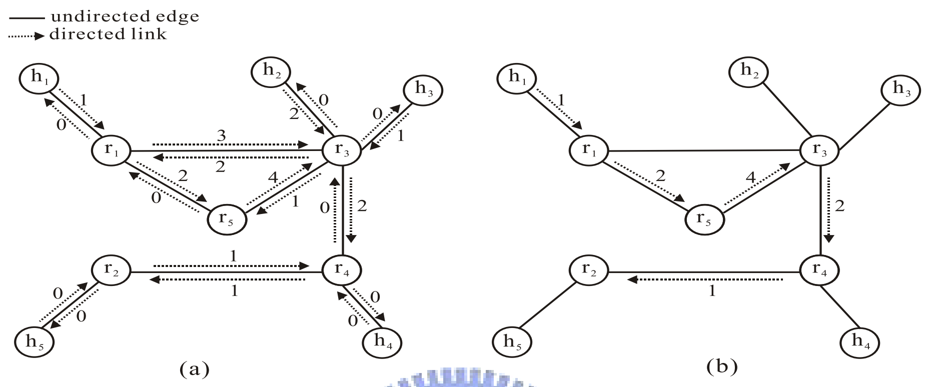


Figure 4.2: (a) shows the identifiers of each directed links and (b) shows only directed links in a sample route.

Although so far we have not given the details for mapping the edge-based representation of a route into a binary bit string, it is clear that the number of bits required for identifying the link in edge-based representation is far smaller than conventional schemes storing all the IP addresses of a path in a network packet. This is based on the fact that in practice a router is unlikely to have a great number of incident links. Thus, the number of bits used to encode a directed link tends to be smaller than the number of bits for storing an IP address. In IPv4 mechanism, an IP address is 32 bits long. To precisely evaluate the number of bits required for the edge-based route representation, we present encoding schemes starting from the most simple, but effective way in terms of space concern.

4.2.2 Encoding Schemes for Edge-based Record Route

To illustrate conversion of the edge-based route representation into a bit string, we first assume that all routers in the network are EBRR-enabled routers. Then, a naive scheme is to use a fixed number of bits to encode link identifiers. In the following context, the set of traversed link identifiers of a packet *p.path.links*, is treated as an integer. *p.path.end* is 32-bit integer. Assume that the globally maximum degree of a router is M. Then, whenever a router receives a packet, it left shifts *p.path.links* by $\lceil \log(M) \rceil$ bits and adds the identifier of the incoming link to the shifted result. Afterwards, it updates *p.path.end* with its IP own address. Algorithm 6 shows the details of the edge-based record router using fixed bits encoding scheme.

Algorithm 6 Edge-Based Record Route Algorithm using Fixed-length Encoding Scheme

```
Encoding Scheme executed at router $r_i$
 1: for each packet $p$ received via link $l_{in}$ do
 2:     $p.path.links \leftarrow p.path.links * 2^{\lceil log(M) \rceil} + id(l_{in})$
 3:     $p.path.end \leftarrow ip(r_i)$.
 4: end for
Path construction scheme executed at the end host
 1: Let $p$ be the attack packet.
 2: Let $r_{curr}$ denote the currently identified router.
 3: $routers(p) \leftarrow \phi$
 4: $E \leftarrow p.path.links$
 5: $r_{curr} \leftarrow p.path.end$
 6: Insert $r_{curr}$ as the first element of $routers(p)$.
 7: repeat
 8:     $id_l \leftarrow E \bmod 2^{\lceil log(M) \rceil}$
 9:     Let $l \in L(r_{curr})$ be the link with $id(l) = id_l$ and $r$ be the router that $l$ is incident from.
10:     Insert $r$ as the first element of $routers(p)$.
11:     $E \leftarrow \lfloor E/2^{\lceil log(M) \rceil} \rfloor$
12:     $r_{curr} \leftarrow r$.
13: until $E = 1$
```

The naive encoding scheme has two fundamental disadvantages. First, in practice, it is difficult to obtain an appropriate value of M. Moreover, the globally maximum degree may dynamically change. This problem cannot be solved by using a "sufficient large" value. The second problem is that using a fixed number of bits to store a link identifier may likely result in unnecessary waste of space. Consider a small network as shown in Fig. 4.2. The global maximum degree is 5 and thus the fixed number of bits for representing a link identifier is 3 ($\lceil log(5) \rceil = 3$). However, it is only necessary for router r_3. For other routers such as r_1, 2 bits is sufficient for encoding its links since the degree of r_1 is 3.

To address this issue, the number of bits for representing a link is determined according to the degree of each router respectively. That is, whenever a router r_i receives a packet p via link l_{in}, it left shifts *p.path.links* by $\lceil \log(deg(r_i)) \rceil$ bits and then add $id(l_{in})$ to the shifted result. In this approach, it is not necessary to obtain the globally maximum degree. Moreover, the number of bits used to represent the whole path is reduced. In fact, based on the same idea, the space efficiency can be further improved. Conceptually, $p.path.links$ is left shifted by $\log(deg(r_i))$ bits rather than $\lceil \log(deg(r_i)) \rceil$ bits. This is achieved by replacing the shift operation with multiplication. That is, when packet p arrives at a router via incoming link l_{in}, $p.path.links = p.path.links * deg(r_i) + id(l_{in})$.

In addition, it is reasonable to assume that in large networks only part of the routers are EBRR-enabled routers, and each EBRR-enabled router is aware of the status of its adjacent routers whether they are EBRR-enabled routers or not. We also assume that an EBRR-enabled router knows the degree of its adjacent routers. With these assumptions, the entire route of a packet is partitioned into a set of *partial paths*, where each partial path contains a successive sequence of EBRR-enabled routers. These partial paths are also referred to as *traceable subpaths*. On the other hand, a path without EBRR-enabled routers are referred to as a *non-traceable subpath*. This change leads to a modification to the original edge-based route representation. The modified definition is given as follows.

Definition 2 (Edge-Based Route Representation with Non-Traceable Subpaths) *The path of a packet p refers to the concatenation of partial paths (or say, traceable subpaths).* $p.path$ *is written as* $(p.path_1, p.path_2, \ldots, p.path_k)$*, where* $p.path_1$ *denote the partial path which is nearest the attack source.* $p.path_i.links$ *represents the sequence of links in a partial path* $p.path_i$*. Similarly* $p.path_i.end$ *denotes the IP address of the last router in the partial path, and* $|p.path|$ *denotes the total number of partial paths.*

Due to the change of assumptions, there is a slight change in the process of edge-based record route. That is, for a partial path $p.path_i$, $p.path_i.end$ is filled with an IP address when packet p is forwarded to a non-EBRR-enabled router. In other cases, only $p.path_i.links$ are updated. Algorithm 7 shows the details of enhanced edge-based record route scheme.

A partial path can be implemented as structure with three fields in the IP header. Fig. 4.3 shows the format of a partial path. It can be divided into three fields. The first field EID (encoded identifier) is used to stored the encoded link identifiers. As we will see in the next section, the length of this field is determined both by the number of EBRR-enabled routers in each partial path and the available space for storing a partial path in

Algorithm 7 Edge-Based Record Route Algorithm using Dynamic-Length Encoding Scheme

Encoding Scheme executed at router r_i

```
1: for each packet $p$ received via link $l_{in}$ do
2:   Let $r_j$ denote the next hop router to which $p$ will be forwarded.
3:   $k \leftarrow |p.path|$
4:   $p.path_{k+1}.links \leftarrow p.path_{k+1}.links * deg(r_i) + id(l_{in})$
5:   if router $r_j$ does not support edge-based record route then
6:     $p.path_{k+1}.end \leftarrow ip(r_i)$
7:     Append $p.path_{k+1}$ as the last element in $p.path$.
8:   end if
9: end for
```

Path construction scheme executed at the end host

```
1: Let $p$ be the attack packet.
2: Let $r_{curr}$ denote the currently identified router.
3: $routers(p) \leftarrow \phi$
4: $i \leftarrow |p.path|$
5: while $i \geq 1$ do
6:   $E \leftarrow p.path_i.links$
7:   $r_{curr} \leftarrow p.path_i.end$
8:   Insert $r_{curr}$ as the first element in $routers(p)$.
9:   repeat
10:     $id_l \leftarrow E \bmod deg(r_{curr})$
11:     Let $l \in L(r_{curr})$ be the link with $id(l) = id_l$ and $r$ be the router that $l$ is incident from.
12:     Insert $r$ as the first element in $routers(p)$.
13:     $E \leftarrow \lfloor E/deg(r_{curr}) \rfloor$.
14:     $r_{curr} \leftarrow r$.
15:   until $E = 1$
16:   $i \leftarrow i - 1$
17: end while
```

an IP packet. The second field is 32 bits long, used to store the end point address of the partial path. The third field, *Pauth*, is used to assure the correctness and authenticity of the partial path. In fact, the Pauth field is used to bind the partial path to the IP packet. It also allows the end host system to validate the source of a partial path. As a result, attackers can neither mislead the path reconstruction procedure by injecting false partial paths. In the next section, we will describe an implementation of the edge-based record route in details, including the field of a packet to store partial paths, the mechanism to deliver partial paths to the end host systems, and the authentication of partial paths.

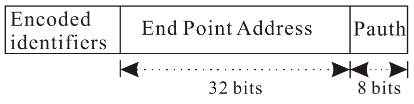


Figure 4.3: The structure of a partial path

4.3 Proposed Solution for Single Packet IP Traceback

In this section, we first present an overview of the proposed scheme. Subsequently, issues regarding the storage of path information in an IP packet, the delivery of partial paths to the destination hosts, and the authentication of a partial path are explored.

4.3.1 Architecture

Figure 4.4 shows the major architectural components of the proposed traceback system. In the architecture, networks composed of directly connected EBRR-enabled routers are organized into a set of *Traceback Enabled Domains* (TED). Networks consisted of legacy routers (i.e. routers that do not support traceback operations) are referred to as *Non-TEDs*. A TED will cooperate with other TEDs to locate the source of an IP packets. In each TED, all routers cooperate to support the operation of edge-based record route. Moreover, there is a Traceback Server (TS) which helps end host systems to construct the route in the TED of a IP packet. Each TS maintains a database of the physical topology of the correspondent TED and manages a set of cryptographic keys which are used to ensure the authenticity of the partial path information. The authentication issues is discussed

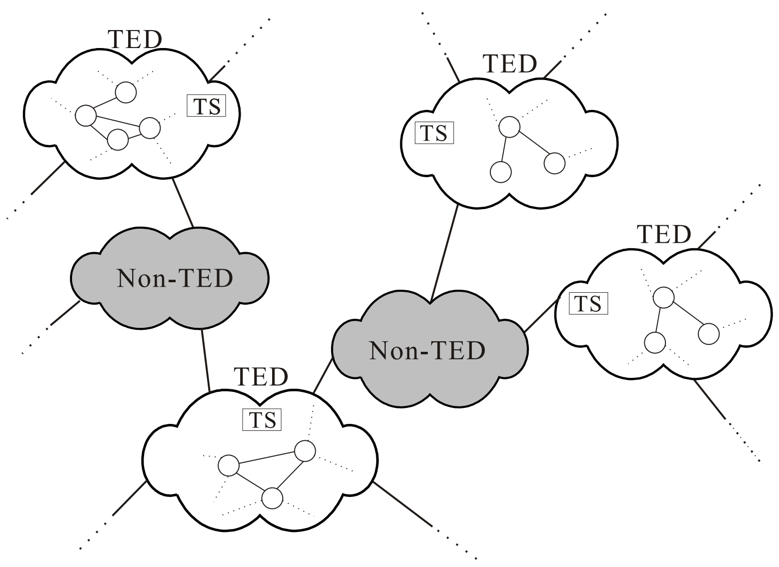


Figure 4.4: The network model

in section 4.3.4. Due to the complex topology of the current Internet infrastructure, there would be several TED distributed over the entire Internet. In this chapter, we do not impose any restriction on the constructions of TEDs except that the TED and TS information of an EBRR-enabled router must be acquired easily. For this purpose, we suggest that these information be stored and accessed via the DNS hierarchy. (Its usage is somewhat like the MX record for locating a mail exchanger of a specific domain.) In addition, as we shall see shortly, in order to authenticate path information, hosts on the Internet, Internet routers and TSes must be loosely time synchronized. This can be easily accomplished via NTP [75], and details of achieving time synchronization is not discussed in the scope of this dissertation.

4.3.2 Operational Overview

Although edge-based route representation allows a route be to encoded using only a small number of bits, delivering the set of partial paths to the destination host remains a fundamental issue that needs to be addressed. One possible solution is to simply store the set of partial paths along with the packet itself. However, it is unreasonable to assume that there is sufficient unused space in the packet. In fact, it is likely that there might be no unused space in an attack packet. Consequently we choose to use an existing IP header field to store a partial path. Then, whenever a partial path is terminated, it is delivered to the destination by a separate packet.

Herein, the definition of a “terminated partial path” is slightly different from the original definition of it. Since no matter which IP filed is selected to store a partial path, the number of bits in the field is limited and thus a partial path may be forced to terminate even the next router is an EBRR-enabled router. Some careful readers may think that using a sufficient large IP field, such as IP Option field of 320 bits long, can reduce the chance that a partial path is forced to terminate due to the space exhaustion. However, this can also increase the probability of mis-identifying the source of an attack if the topology of the Internet changes frequently. Consider that a new link (or say, network interface) is added to a router that has been record in a partial path. Consequently, the number of links of the router would increase by one. It is clear that the source identified by the partial path will be incorrect due to the change of the topology. Since the probability of mis-identifying the source increases as the number of routers recorded in a partial path gets larger. To reduce the possibility of errors, the number of routers of a partial path should be restricted in a reasonable manner. For the reliability reason and practice concern, we choose 16-bit length for the encoded link identifiers of a partial path. This is partly because we can find a 16-bit IP header field for storing the encoded link identifiers.

In our approach, the 16-bit checksum field is overloaded to store the encoded link identifiers of a partial path and ICMP packets are employed to deliver partial paths to destination hosts. The original function of the checksum field is to ensure the integrity of the IP header fields. It is selected due to the following three observations. First, routers are usually connected to each other using high quality physical media. Thus, the transmission error rate between two adjacency routers is low. Second, the error detection/correction mechanism in the data link layer can further reduce the occurrence of errors. Third, upper layer protocols, such as TCP and UDP, include several IP header fields for computing their own checksum. (The checksum field of IP header is excluded.) This helps ensure the integrity of important IP header fields. Thus, the function of the checksum field is considered replicated and can be overloaded without greatly affecting network infrastructure, protocols, and existing applications. IP protocol does not need to be revised. Note that any such overloading may sacrifice certain function of the overloaded field and although the checksum field is overloaded in our scheme, our solution here only demonstrates a possible design rather than attempts to find the optimal balance among many potential implementation trade-offs.

As mentioned above, since there are only 16 bits available in the checksum field, it cannot accommodate a route with unlimited number of hops. In this case, a partial path is forced to be terminated even when the next router is an EBRR-enabled router. More specifically, a partial path is forced to be terminated if a router finds that there is no sufficient space for the next router to inscribe the incoming link identifier information in the checksum field. This takes place when the value in the checksum field becomes greater than 2^{16}. Consider a packet p arrives at router r_i via link l_{in} and the next hop router to which p will be forwarded is another EBRR-enable router r_j. After r_i inscribes the link identifier $id(l_{in})$ into the checksum field , r_i should also examine whether r_j can inscribe the next incoming link identifier into the checksum field or not. Let $p.checksum$ denote the value of the checksum field. If $(p.checksum + 1) * deg(r_j) \geq 2^{16}$, the current partial path is forced to be terminated and is delivered to the destination host via an ICMP packet. After that, the checksum field is filled with a designated value to denote the beginning of a new partial path.

It is worthy to note that the checksum field is overloaded only when an IP packet is transmitted in a partial path. In other words, by correctly configuring EBRR-enabled routers, overloading the checksum field has the advantage of being backward compatible with legacy routers due to the fact that the checksum field can be reloaded with conventional checksum value. In short, the function of the checksum field is reloaded when the packet is forwarded to a router that does not support edge-based record route. To meet the requirement, the last EBRR-enabled router has to calculate and reload the checksum field with a value that is consistent with the specification of IP protocol before the packet is forwarded to a EBRR-disabled router. Otherwise, the packet will be discarded due to the occurrence of checksum error. It is worthy to note that the destination host will always receive a packet with conventional checksum value because the last EBRR-enabled router, no matter it is the last hop router of entire path or not, will restore the checksum value, and generate an ICMP packet for partial path delivery, namely *PPD packets*. On the other hand, when an EBRR-enabled router receives an IP packet from an upstream router that is not EBRR-enabled, it first examines the integrity of the IP packet header. If the packet has a valid checksum, the EBRR-enabled router will set the checksum to a designated value, and then perform subsequent processing. If the the checksum is invalid, the packet is discarded.

Whenever a partial path is terminated or forced to be terminated, a PPD packet is generated for delivering a partial path to the destination host. If a large number of PPD packets are generated for partial path delivery, they may seemingly impose substantial overhead to both the routers and the network. To reduce the overhead, we choose to send partial paths in a batch — several partial paths are delivered to the destination host using one PPD packet. This is accomplished by storing partial paths in the IP option field first. Since the IP option field is now used only for testing and debugging, for most IP packets this space is left unused. Therefore, by taking advantage of the IP option field, several partial paths can be temporarily stored here and be delivered to the destination host in one PPD packet. The IP option field can accommodate at most 5 partial paths. Consequently an PPD packet is created after five partial paths are collected in the IP option field. (Recall that the encoded identifiers of a partial path is of the same length as a checksum field, that is, 16 bits long. The end point address field is a 32-bit field and the Pauth field takes 8 bits. Thus, 7 bytes is required for storing each partial path. Since the IP options field is 40 bytes long, the IP option field can accommodate a total of five ($\lfloor 40/7 \rfloor = 5$) partial paths.) In other words, when unused space in the IP option field becomes smaller than 7 bytes, the set of partial paths in the IP option field are sent by one PPD packet. Afterwards, the IP option field is cleared so that subsequent partial paths can be stored in this field.

Note that the IP option may not be available for storing the partial paths if any of existing IP option, such as Source Route, TimeStamp, or Record Route [90] is set. In this case, a partial path is delivered to the destination host by a separate PPD packet every time when it is terminated or forced to be terminated. Algorithm 8 shows the details of edge-based record route algorithm.

Consider the case that the IP option field is available for storing the partial paths. In our approach, the IP option is formatted as shown in Fig. 4.5. The code field is filled with 97 (which indicates option class 3 and the option number1). The pointer field specifies the offset within the option to the next available slot for storing a partial path. The initial value of pointer field is 2. Before moving a currently terminated partial path to the option field, a router first examines the pointer. If the pointer value is less than 33, the partial path is appended at the position specified by the pointer and then the

Algorithm 8 Edge-Based Record Route executed at router r_i

```
1: for each packet $p$, which is received via incoming link $l_{in}$ do
2:   Let $p.checksum$ denote the current value of the checksum field in the packet header.
3:   Let $r_{prev}$ denote the router connecting to $r_i$ via the link $l_{in}$
4:   Let $r_{next}$ denote the next router to which packet $p$ will be forwarded.
5:   if $r_{prev}$ does not support edge-based record route then
6:     $p.checksum \leftarrow 1$
7:   end if
8:   $p.checksum \leftarrow p.checksum * deg(r_i) + id(l_{in})$
9:   if $((p.checksum + 1) * deg(r_{next}) \geq 2^{16}$ or $r_{next}$ is not a EBRR-enabled router then
10:    Terminated the current partial path and denoted as $p.path_x$
11:    $p.checksum \leftarrow 1$
12:    if there is any existing IP option set then
13:      Send an PPD packet that carries $p.path_x$ to the destination host
14:    else
15:      Append the $p.path_x$ to the IP option field
16:      if unused space in the IP option field $< 7$ bytes then
17:        Send an PPD packet that carries the set of partial paths stored in the IP option
           to destination host
18:        Clear the IP option field
19:      end if
20:    end if
21:  end if
22:  if $r_{next}$ is not a EBRR-enabled router then
23:    Compute conventional checksum value and store it in the checksum field.
24:  end if
25: end for
```

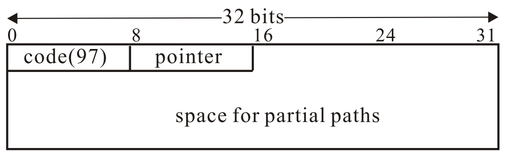


Figure 4.5: The arrangement of IP option for storing partial paths

pointer is increased by 7. Otherwise, the option field is said to be fulfilled and an PPD packet is used to carry the set of partial paths in the option field to the destination host. Afterwards, the partial path is placed as the first partial path in the IP option field and the pointer is set to 9.

4.3.3 Generation of PPD Packets

Using separate packets to deliver partial paths raises several issues. First, an end host system must be able to identify the set of PPD packets which carry partial paths of a particular IP packet. Since an end host may receive a number of PPD packets, there should be an association between an IP packet and the set of PPD packets carrying its partial paths. Second, an end host system must be able to authenticate the sources of PPD packets. Since an attack can also generate PPD packets carrying a set of partial paths to a victim, end host system must be able to authenticate the received PPD packets and their association. Third, an end host system must be able to determine the order among the set of authenticated PPD packets which are associated with the same IP packet.

To address the first issue, each PPD packet contains a digest value used to relate the PPD packet to a particular IP packet. Since the digest is used to uniquely identify a IP packet, its value in PPD packets associated with the same IP packet must be identical. This digest value is computed by applying a hash function to selected IP packet header fields of the packet. With careful selection of header fields, the end host can effectively identify a set of PPD packets associated with an IP packet. The invariant IP header fields are selected as inputs to a digesting function. As shown in Fig. 4.6, marked IP fields are used to generate the digest value.

As to the second issue, each PPD packet has a 32-bit Iauth field which is presented in Section 4.3.4 in further details. Finally, as to the third issue, each PPD packet has a distance value which is used to determine its order. This distance value is obtained from

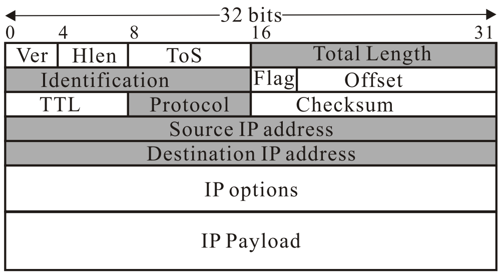


Figure 4.6: Selected IP fields for computing the digest value

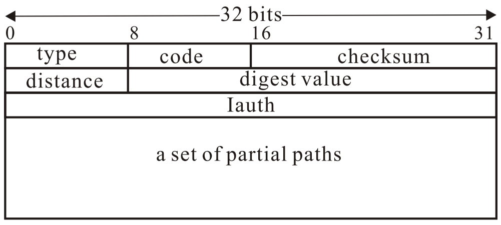


Figure 4.7: The format of a PPD packet which is used to transmit partial paths to end hosts

the TTL value of the IP packet. Specifically, whenever a router decides to deliver a set of partial paths to the destination host by an PPD packet, in addition to the computing of the digest value, it copies the current TTL value to the distance field in the PPD packet. The entire PPD packet format is depicted in Fig. 4.7.

4.3.4 Authentication

Authentication is an essential component in the proposed scheme. It allows an end host to filter out false path information and subsequently helps to correctly identify the source of an attack. In our approach, partial paths are treated as basic units that constitute the entire path. Thus, as we have seen in Fig. 4.3, there is an one-byte "Pauth" field which is used to assure the authenticity of the partial path. Furthermore, PPD messages are used to carry partial paths to the end host systems. It is also important to authenticate PPD message so as to prevent the attack from injecting forged PPD packets to the victim. Herein, we present the way to generate the "Pauth" and "Iauth" field respectively.

The "Pauth" field can be generated in several ways. One straightforward approach is to have the last router in a partial path generates a digital signature using public key cryptography. However due to the high computational overhead of public key cryptographic operations, this approach is infeasible in many network environments. Thus, we choose conventional cryptographic system which may incur less computational overhead. We adopt one-way key chain to achieve lightweight authentication. The technique is originally proposed to authenticate the sources of messages in a broadcast/multicast network environment [88,89]. One of its important characteristics is the use of symmetric cryptography that perfectly fits the lightweight requirement of generating signatures for partial paths. In the following, we detail the use of one-way key chain to achieve authentication.

In the one-way key chain framework, we assume that that EBRR-enabled routers, TSs, and end-hosts are loosely time synchronized. That is, their clocks are time synchronized up to a maximum error of Δ. Moreover, the maximum propagation delay of an IP packet from source to destination is denoted as T_{prop}. Then, the time is divided into a set of uniform intervals, and each interval is of duration T_{int} and $T_{int} \gg \Delta$, Time interval 0 starts at time T_0 and time interval i starts at time $T_i = T_0 + i * T_{int}$. Each EBRR-enabled router constructs a reversed one-way key chain and shares its key chain with the TS in its correspondent TED. That is, in a key chain, each time interval is associated with a secret key which is used to generate signatures in the corresponding time period. Each secret key can then be disclosed after a small amount of time after the key expires. Specifically, consider a key K_i which is associated with time interval $[T_i, T_{i+1}]$. K_i can be disclosed at time $T_i + T_{int} + T_\delta$ where T_δ represents the waiting time after K_i expires and $T_\delta \gg T_{prop}$. The set of keys are organized as a reverse key chain sequence, as shown in Fig. 4.8, that is, $K_i = H(K_{i+1})$. Since the key is disclosed reversely, the disclosure of K_i does not reveal K_{i+1}.

As Fig. 4.8 shows, a key chain can be generated in a straightforward manner. An EBRR-enabled router first randomly generates a random number as K_L (here we assume the length of key chains is a pre-defined parameter within a TED). Then, the subsequent elements of the key chain can then be generated. To allow a TS to generate the same key chain, the router securely sends K_L to the TS and consequently the TS can then construct a key chain that is identical to the key chain in the EBRR-enabled router. Whenever a key chain is exhausted (all of the secret keys have expired), each EBRR-enabled should reconstruct a new key chain and deliver K_L to TS for generating a new key chain in TS. Note that, we do not impose any restriction on the length of the key chain. The settings of T_{int} and T_δ are not strictly limited. They can be set to an hour, twelve hours, a day or even a week. However, setting a "too-long" duration for T_{int} and T_δ will delay the time

$$\begin{tikzcd}K_0 & K_1 \arrow[l, "H(K_1)"'] & K_2 \arrow[l, "H(K_2)"'] & \cdots \arrow[l] & K_{L-2} & K_{L-1} \arrow[l, "H(K_{L-1})"'] & K_L \arrow[l, "H(K_L)"']\end{tikzcd}$$

Figure 4.8: Reversed one-way key chain

for the disclosure of secret keys and consequently delay the time to start the traceback process since each partial paths must be authenticated first. Thus, we suggest that T_{int} be set to an hour and T_δ to 30 minutes. (This suggestion comes from the observation that $\Delta \ll$one hour and $T_{prop} \ll 30$ minutes)

Whenever a router decides to terminate a partial path, denoted as $p.path_x$, the router will compute the value of the Pauth filed by applying a keyed hash function to a partial path and some other fields in the IP header. Specifically, let K_i denote the key of the router in the correspondent time interval when the partial path is to be terminated, and ζ denote the concatenation of values of the IP header fields which are selected for computing the digest values, as depicted in Fig. 4.6. Then $Pauth = H_1(p.path_x.links, p.path_x.end, \zeta, K_i)$, where $H_1(.)$ denotes a HMAC [11,65] function that generates one-byte keyed hash result. In this way, the victim can verify the authenticity of each partial path by examining the "Pauth" field once the correspondent secret key of the router is disclosed. Shortly we will present how the victim can acquire required secret for authenticating partial paths. In this way, as we shall see in Section 4.4.2, attackers can only have the probability of 1/256 to mislead the victim by forging an partial path or replay any partial path that he has recently accumulated.

Similar to the generation of the "Pauth" field, the "Iauth" field is also generated by applying a hash function to some of IP and PPD header fields, and the set of partial paths in the PPD message. That is, $Iauth$=$H_2(ip_{src}$,ip_{dst},distance, digest value, partial paths in the PPD message, K_i), where ip_{src} and ip_{dst} denote the source and destination IP address of the PPD packet, K_i denote the key of the current router that generates this PPD packet and $H_2(.)$ is a HMAC function that generates a four-byte keyed hash result.

For a victim who wishes to authenticate a partial path (or an PPD packet) received at time $T_{arrival}$ and within the duration of T_i, he must acquire correct keys first. The victim will first lookup the TS associated with the router R which terminated the partial path (or delivering the PPD packet) via DNS hierarchy. Afterwards, the victim waits until T_i+T_{int}+T_δ+Δ, and then send a key retrieval request to the TS. The request indicates the time that the offensive packet arrives at (i.e. $T_{arrival}$) and the router R. After receiving the request, the TS can retrieve the key chain of R. Subsequently, TS returns secret keys which fall in the duration between $T_{arrival} - \Delta + T_{prop}$ and $T_{arrival} + \Delta + T_{prop}$. (Since $T_{int} \ll \Delta$, the victim may receive two secret keys when $T_{arrival}$ is near the boundary of two consecutive intervals. In most cases, only one secret key is returned from TS.) Finally, the victim can authenticate the partial path (or PPD packet) using the retrieved keys. If there exists a secret key that can successfully authenticate the partial path (or PPD packet), it is considered valid. Otherwise, the partial path (or PPD packet) will be dropped.

4.3.5 Traceback Processing

The traceback process is triggered by the detection of the presence of an attack packet. Given an attack packet, a set of PPD packets carrying the partial paths of the attack packet can be identified by the digest value. Subsequently, the order of these PPD packets is then determined according their distance values. Partial paths stored in these PPD packets are then collected and sorted. (Note that partial paths stored in the same PPD packet are stored sequentially.) Then, for each partial path, the victim can identify the TS which is indicated in the end point address field of the partial path. Afterwards, the victim can acquire a key from the TS to authenticate the partial path. In this way, the victim can construct a sequence of authenticated partial paths which are denoted as ($p.path_1$,$p.path_2$, ..., $p.path_k$), where $p.path_1$ represents the candidate partial path that is nearest the source.

To extract the subpath encoded in $p.path_i$, the victim send $p.path_i$ to the TS associated with $p.path_i.end$. The TS can then construct the subpath encoded in the partial path. It is also possible that one partial path may encode a traceable subpath which across more than one TED. In this case, since the current TS can identify the TS the upstream router is associated with, the current TS can deliver necessary part of the partial path to the new TS and then can acquire the upstream attack path. After the attack subpath encoded in $p.path_i$ is constructed, it is returned to the victim.

It is worthy to note that $p.path_1$ is sufficient for locating the source of an attack. In other words, the source can be identified by constructing the subpath encoded in $p.path_1$. Pseudo code for constructing the entire attack path is shown in Algorithm 9.

Algorithm 9 Pseudo code of the traceback process

```
1: Let $p.path = (p.path_1, p.path_1, \ldots, p.path_k)$ denote the sequence of authenticated partial
   paths retrieved from the set of PPD packets.
2: Let $r_{curr}$ denote the currently identified router.
3: $routers(p) \leftarrow \phi$
4: $i \leftarrow |p.path|$.
5: while $i \geq 1$ do
6:    $E \leftarrow p.path_i.links$
7:    $r_{curr} \leftarrow p.path_i.end$
8:    Let $TS_{curr}$ denote the current TS associated with $r_{curr}$
9:    Send $p.path_i$ to $TS_{curr}$ and let $r_{list}$ denote the sequence of routers returned from $TS_{curr}$.
10:   Insert $r_{list}$ in the beginning of $routers(p)$.
11:   $i \leftarrow i - 1$
12: end while
13: Output the first element in $routers(p)$ as the candidate source of the attack
```

4.4 Analysis

In this section, two important characteristics of the proposed edge-based record route scheme are analyzed. First, overhead involving in tracing the origin of a single attack packet is estimated. This overhead is measured in terms of the number of ICMP packets generated for carrying partial paths. Second, the accuracy of our approach is evaluated.

4.4.1 Overhead Analysis

The cost for tracing the source of a single packet heavily relies on the overhead for generating PPD packets which carry partial paths. This overhead can be measured in terms of the number of PPD packets generated. According to our previous description, the number of PPD packets heavily depends on the number of partial paths in a route. A partial path is formed each time the checksum field is moved to the IP option field (if possible) or is delivered to the destination host via an PPD packet. In the former case, the number of PPD packets can be obtained by dividing the number of partial paths by five, which is the number of partial paths IP option field can accommodate. In the latter case, the number of PPD packets equals to the number of partial paths since each each partial path requires an PPD packet.

It is necessary to estimate the number of partial paths before computing the number of PPD packets. In order to do this, there are several basic assumptions must be presented. First, the degree of each router is assumed to be independent of each other, and we use α to represent the expected number of bits for a router to encode its links. Here, we first consider the simplest case, that is, all the routers in the path are EBRR-enable routers. Let $f(L)$ denote the number of partial paths, where L is the number of routers in the path. Then, $f(L)$ can be expressed as:

$$f(L) = \lceil L * \alpha/16 \rceil$$

Next, consider general cases such that not all of the routers are EBRR-enabled routers. To calculate the expected number of partial path of a route containing L routers, we first divide the entire route into k segments, $1 \leq k \leq L$. The lengths of segments is represented as a sequence of integers and denoted as $S^{L,k}$=($S_1^{L,k}$, $S_2^{L,k}$, $\ldots$, $S_k^{L,k}$) such that $S_i^{L,k} \geq 1$ for $1 \leq i \leq k$ and $L = \sum_{i=1}^{k} S_i^{L,k}$. Let $L_{evensum}^{S_i^{L,k}} = \sum_{i=1}^{\lfloor k/2 \rfloor} S_{2*i}^{L,k}$ and $L_{oddsum}^{S_i^{L,k}} = \sum_{i=0}^{\lfloor k/2 \rfloor} S_{2*i+1}^{L,k}$. Then, Let $\psi(S^{L,k}) = \sum_{i=1}^{k} f(S_i^{L,k})$, $\psi_{odd}(S^{L,k}) = \sum_{i=0}^{\lfloor k/2 \rfloor} f(S_{2*i+1}^{L,k})$ and $\psi_{even}(S^{L,k}) = \sum_{i=1}^{\lfloor k/2 \rfloor} f(S_{2*i}^{L,k})$

Under the same L, k constraint, there are in total C_{k-1}^{L-1} such sequences and let $A^{L,k}$ denote the set of those sequences. In addition, we use $A_i^{L,k}$ to denote an element of $A^{L,k}$. Then, consider a path containing L routers. $F(L)$ denotes the expected number of partial paths. $F(L)$ can be computed as:

Next, consider a simple case that each router are administrated independently and that EBRR-enabled routers are randomly distributed on the Internet. In this case, let p denote the probability that an IP packet will arrive at a router that is EBRR-enabled. $F(L)$ denotes the expected number of partial paths. And, in this simple probabilistic model, $F(L)$ can be expressed as:

$$F(L) = \sum_{k=1}^{L} \sum_{i=1}^{C_{k-1}^{L-1}} p^{L_{oddsum}^{A_i^{L,k}}} * (1-p)^{L_{evensum}^{A_i^{L,k}}} (\psi_{odd}(A_i^{L,k})) + p^{L_{evensum}^{A_i^{L,k}}} * (1-p)^{L_{oddsum}^{A_i^{L,k}}} (\psi_{even}(A_i^{L,k}))$$

Consider a more complicated case that Internet routers are managed by serveral administrators. That is, Internet routers support the proposed scheme in a consistent manner. We assume that routers connecting to each other are likely to reside in the same Autonomous System (AS). Since routers in the same AS tend to be administrated under the same policy, they will support the edge-based record route in a consistent manner. That is, routers connecting to an EBRR-enabled router tend to be EBRR-enabled routers and vice versa. This assumption can be modeled by a state-transition diagram, as shown in Fig 4.9. In the figure, S_1 denotes the state that the current router is EBRR-enabled, and S_2 denotes the state that the current router is not EBRR-enabled. A packet is of probability q to remain in the same state when it moves to the next router, and is of probability $(1-q)$ to transit to the other state. The probability that the first router is an EBRR-enabled router is set to 1/2, and on the other hand the probability for not being an EBRR-enabled router is 1/2.

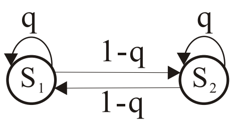


Figure 4.9: A probability model that reflects the state transition of IP packets.

In the probabilistic model presented above, $F(L)$ can be computed as:

$$F(L) = \sum_{k=1}^{L} 1/2 * q^{L-k} * (1-q)^{k-1} * (\sum_{i=1}^{C_{k-1}^{L-1}} \psi(A_i^{L,k}))$$

Fig. 4.10 shows how the probability q and the value L affect $F(L)$ when α is fixed at 16. The intuition is that $F(L)$ is linear with respect to L, and q determines the slope. A smaller q results in a greater slope.

Notice that in cases of $q > 0.5$, $F(L)$ grows greater than 5 when L is about 20. This indicates that when a packet traverse more than 20 hops, five partial paths are collected. However, this case occurs rarely. It is reported that the average hop count of current network flows is about 15 and only a few network flows traverse more than 20 hops [32, 54]. This observation implies that if the IP option field can be used to store partial paths, the number of PPD packets can be greatly reduced because only a few IP packets would have more than 5 partial paths. In other words, the number of PPD packets tends to be small. On the other aspect, the number of PPD packet would increase by one when the IP packet traverse for more 20 hops. This indicates that IP packets traverse less than 40 hops would have PPD packet less than two.

Next, we examine the value of $F(L)$ on different α values. Fig. 4.11 shows that α does not have a great impact on $F(L)$. According to the figure, α is another factor that can affect the slope. The smaller α is, the lower the slope is. However, the difference is small. This experimental result also shows that, for IP packets that traverse fewer than 20 hops, the number of partial paths is smaller than five. Consequently, no PPD packet will be generated in this case.

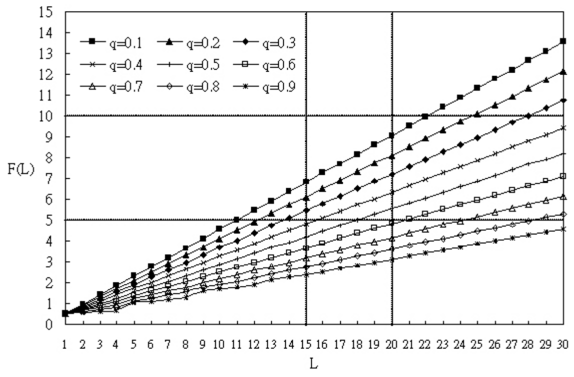


Figure 4.10: How L and q affect $F(L)$ when $\alpha = 4$

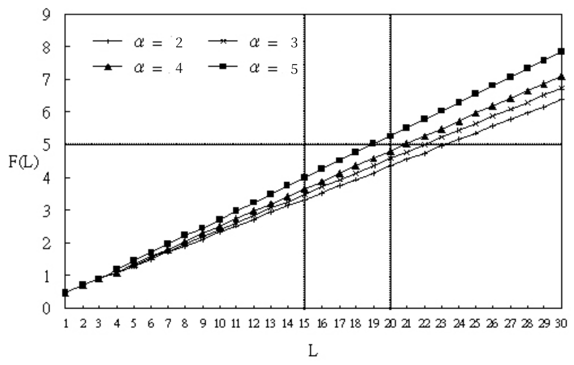


Figure 4.11: How L and α affect $F(L)$ when $q = 0.6$

In summary, our experimental results show that, in general cases, an IP packet would arrive at the destination host along with a complete and ordered set of partial paths. Thus, the overhead for tracing the origin of a single IP packet is small. But in the case that an IP packet is sent with an IP option set, the number of PPD packet will be equal to the number of partial paths in the route. This implies that the attackers would always let an IP option set in the attack packet to increase the cost for back tracing. Nevertheless, this cost is still relatively low when the increased number of PPD packets is compared to the total number of packets in the network.

In addition, readers might also think that the edge-based record route would provide (D)DoS attackers with a form of amplification for their attacks since the number of packets that the victim receives would increase due to the increase of PPD packets. This is correct. However from the perspective of preventing network attacks (including DDoS attacks), our approach can effectively deter these attacks since the origins of the attacks would be discovered easily. From this point of view, even in the case of (D)DoS attacks, it is easy to stop attacks at their sources once the attacks are detected.

4.4.2 Accuracy Analysis

The accuracy of the identified source of an attack is highly related to the set of candidate partial paths which are viewed as associated with the attack packet. When this set of partial paths is incorrect, the proposed scheme may return a false origin of the attack flow. This occurs when there exists at least one packet which has the same digest value with the attack packet. In addition, according to our previous experimental results, the packet must traverse more than 20 hops. Otherwise, there will no PPD generated for the packet.

To evaluate the accuracy, let T_{det} denote the time (in seconds) that an IDS needs to determine whether a packet is intrusive or not, and λ denote the packet arrival rate (number of packets per second) of the destination host. We assume that PPD packets associated with an IP packet will be received in a small amount of time later than the time the packet itself arrives at the destination host. We also assume that an attack packet will have an existing IP option set. In this case, the number of PPD packets created for the attack packet will be equal to the number of partial paths in the route.

Here, we first estimate the accuracy of our scheme with the presence of hash collisions.

In our scheme, the hash collision rate is is 2^{-24}, and there will be at most $(\lambda * T_{det} - Z)$ PPD packets. In other words, the expected number of collisions would be $((\lambda * T_{det} - Z) * 2^{-24})$. Our scheme can identify the correct source if the distance values in all collided PPD packets are smaller than the biggest distance value of correct PPD packets. (Notice that the biggest distance value is greater than $2Z - 1$.) In other words, the accuracy can be expressed as:

$$((2Z-1)/256)^{(\lambda * T_{det} - Z) * 2^{-24}}$$

Consider a victim host with 100 Mbps ethernet links. The maximum packet arrival rate is about 250K pps (packet per second) if all packets are about 50-byte long. In addition, assume that the average detection time for the deployed IDS system to capture an attack packet is 1 second. Then, the accuracy of our scheme will be greater than 0.9998. That is, the accuracy of the proposed scheme in this case is about 99.98%

Next, consider an attacker that forge PPD packets in order to deceive the victim from identifying a wrong attack source. Notice that each PPD packet needs to be authenticated via its Iauth field, and a partial path carried by an PPD packet can be accepted if both the Iauth field in the PPD packet and the Pauth filed in the partial path are valid. Notice that the last partial path record in a PPD packet is created by the EBRR-enabled router which generates the PPD packets. Therefore, the Pauth field of the last partial path and the Iauth field of the PPD packet are created using the same singing key. Thus it is reasonable to assume that if an attacker attacker can successfully forge a 32-bit Iauth field, then he/she would be able to successfully forge the Pauth field. Since an attacker does not have the signing keys which associated with EBRR-enabled routers, the best an attacker can do is to randomly guess the Iauth value (the probability of a successful guess is 2^{-32}), and in this case the accuracy of our scheme can be estimate as:

$$(1 - 2^{-32})^{\lambda * T_{det} - Z} \approx 1 - 2^{-32} * (\lambda * T_{det} - Z)$$

In this case, the accuracy of our scheme will be greater than $1 - 2^{-32} * (2^{18} * 1 - 0) = 1 - 2^{-14} \approx 0.99987$. In other words, the accuracy of our scheme under this attack is about 99.987%

Next, we examine the robustness of the proposed scheme under another attack. That is, the attacker pre-filled a forged partial path in the IP Option field. In this case, since the

Pauth field is only 8-bit long, the attack has much higher probability, 1/256, to mislead the traceback process. One of the straightforward way to reduce the probability is to allocate more bits for the Pauth field for each partial path. However, this would increase the size of a partial path and consequently increase the number of PPD packets. So far, it is believed that a 8-bit Pauth field is sufficient since the proposed scheme have the probability of 255/256 (about 99.6%) to correctly identify the attack source. Overally, it is difficult for attackers to deceive the traceback process from identifying a wrong attack source.

4.5 Summary

Developing a traceback system for discovering the origin of a single packet is considered a very difficult problem. In this chapter, we proposed the edge-based record route scheme to cope with it. Our solution shows that single packet tracing can be accomplished with high accuracy at low cost. Edge-based record route has the following characteristics:

- The traceback process can be performed after an attack occurs.
- It does not require storage in routers. The path information of IP packets are delivered to the destination hosts via the packet itself or PPD packets. This can significantly reduce the difficulty of deployment. The success of a traceback process is not affected by the status of routers. Even if a router crashes when the traceback starts, it can still accurately identify the attack source.
- The traceback process does not require interaction with routers in the attack path. This significantly reduces the complexity of the traceback scheme.
- The proposed scheme can be incrementally deployed. Its usefulness greatly increases with respect to the scale of deployment. It can construct portions of the attack path whenever there are EBRR-enabled routers.
- The authentication mechanism allows a victim to filter out false path information which is generated by attackers.
- The overhead for tracing individual packets is small. Only a few IP packets will trigger the generation of PPD packets. Most IP packets will arrive at the destination

hosts with an ordered and complete list of partial paths stored in the IP option field.

- Finally, the accuracy of the proposed scheme is high. Our analysis indicates that the proposed scheme has high probability to accurately locate the origin of attack packets.

Although edge-based record route is effective and efficient, there are still issues which need further investigation. One of the challenges is to reduce unnecessary packet fragmentation which is triggered by the insertion of 40-byte IP option field into the IP packet header. This overhead cannot be ignored in some environments. Several other issues remain unsolved, such as evaluating the effectiveness under distributed attacks and improving the robustness when there are broken links in the attack path. As a whole, we believe that our solution represents an important step toward an automated traceback facility that can be used to trace the origin of a single IP packet.

Chapter 5

A Source-End Defense System for DDoS Attacks

In this chapter, a DDoS defense scheme is proposed to deploy in routers serving as the default gateways of sub-networks. Each router is configured with the set of IP addresses belonging to monitored sub-networks. By monitoring two-way connections between the policed set of IP addresses and the rest of the Internet, our approach can effectively identify malicious network flows constituting DDoS attacks, and consequently restrict attack traffics with rate-limiting techniques. Current source-end DDoS defense scheme cannot accurately distinguish between network congestion caused by a DDoS attack and that caused by regular events. Under some circumstances, both false positive and false negative can be high, and this reduces the effectiveness of the defense mechanism. To improve the effectiveness, new DDoS detection algorithms are presented in this chapter to complement, rather than replace existing source-end DDoS defense systems. The design of the proposed detection algorithm is based on three essential characteristics of DDoS attacks: *distribution*, *congestion*, and *continuity*. With the three characteristics, the proposed detection algorithm significantly improves detection accuracy, and at the same time reduces both false positive and false negative against DDoS attacks.

5.1 Preliminaries

Current Internet infrastructure is vulnerable to network attacks, and particularly, many security incidents have shown that the Internet is weak against distributed denial-of-service (DDoS) attacks. In general, a DDoS attack is accomplished by persistently overloading critical resources of the target Internet service so as to completely disable or degrade the service over an extended period of time. Such resource overloading can be achieved in several ways. First, an Internet service can be overloaded by a large number of service requests issued in a short period of time. As a result, legitimate service requests may be dropped due to insufficient resource such as computation power or memory space. Second, attackers can overload a network link near the target, and consequently, all flows traverse through the link will experience significant degrade of service quality.

To generate a great amount of traffic or service requests, attackers may first compromise a large number of computer systems. This can be easily accomplished due to the large number of insecure computer systems and the set of easily acquired and deployed exploit programs, such as Tribal Flood Network (TFN), TFN2K and Trinoo. On the other hand, detecting or preventing a DDoS attack is relatively much harder. The lack of explicit attack signatures/patterns makes it extremely difficult to distinguish attacks from legitimate traffic. Furthermore, the anonymous nature of IP protocol allows the attackers to disguise the attack origins, and thus makes it hard to detect the sources of DDoS attacks. These difficulties make the construction of an effective DDoS defense mechanism become a very challenging problem.

Issues for defending DDoS attacks have been extensively investigated in recent years and existing approaches have been discussed in Chapter 2. Recall that defending against DDoS attacks at their sources is a very attractive idea; however, detecting the occurance of a DDoS attack at attack sources is also a very difficult task. [28]. The main difficulty arises from the insignificant aggregate of attack traffic which can be observed in attack sources. Other criteria for identifying DDoS attacks must be discovered. For instance, in the D-WARD system proposed by *Mirkovic et al* [76], network congestion measured by the ratio of incoming and outgoing packets of network connections is used to judge whether the monitored flow is part of a DDoS attack or not. By monitoring the changes of the ratio, D-WARD would be able to detect a DDoS attack that has already disable the victim. However, it is hard for D-WARD to distinguish a DDoS attack from network congestion caused by other events. On one hand, D-WARD can mis-classified a flow if the ratio of flow is high in its normal operation. On the other hand, D-WARD is weak in detecting low-rate attacks. In other words, a well-designed attack script can avoid being detected by D-WARD by carefully control the congestion caused by the attack.

To address the weakness of D-WARD, in this chapter, we propose a source-end DDoS detection algorithm and an attack response mechanism, where the former can accurately identify an ongoing DDoS attack and the latter can effectively limit attack traffic in source networks. The proposed detection and response algorithms are built upon the system architecture originally proposed in D-WARD. Our proposal focuses on reducing both false positive and false negative on detecting two-way connections. That is, the proposed scheme attempts to complement, rather than replace the D-WARD system.

The design of proposed scheme is based on the observation of three essential characteristics of a DDoS attack: *distribution*, *congestion*, and *continuity*. *Distribution* refers to the spreading of attack traffic from a large number of compromised hosts. *Congestion* refers to the inherent consequence of a DDoS attack. That is, an increasing packet loss rate observed in a monitored network flow would generally represent a signal of a DDoS attack . Third, *continuity* directs to the observation that network congestion caused by DDoS attacks usually lasts for an extended period of time. Combining the above three criteria allows us to differentiate a DDoS attack from a typical network congestion caused by other events. Based on the three characteristics, a new DDoS defense mechanism is proposed. Since the proposed mechanism is built upon D-WARD architecture, the proposed DDoS defense mechanism is also deployed at routers serving as the default gateways. On-line traffic statistics, in terms of distribution, congestion, and continuity, are gathered and compared against previous statistics derived from normal traffic. In this way, malicious network flows are identified and rate-limited. Rate limits are dynamically adjusted according to the behavior of malicious network flows. On one hand, dynamic adjustment allows a misclassified network flow to regain network bandwidth when the flow shows compliance to legitimate flow model. On the other hand, since attack scripts has no way to distinguish the effect of rate-limiting from that of a successful DDoS attack, dynamic adjustment helps restrain malicious flows.

This chapter is organized as follows. Section 5.2 gives an review of the D-WARD system. The proposed source-end DDoS defense scheme is presented in Section 5.3, including its detection and rate-limiting mechanism. Section 5.4 describes an implementation of the proposed scheme and presents several experiments on estimating the effectiveness. Subsequently, we summarize and conclude our findings in Section 5.5.

5.2 Review of D-WARD

In this section, we briefly review D-WARD system, including system architecture, detection algorithm, and attack response algorithm.

5.2.1 System Architecture

From the architectural point of view, D-WARD consists of a *observation component* and a *throttling component*. The observation component examines all communications between the set of IP addresses in the monitored network and the external IP addresses, and then computes on-line traffic statistics. Note that, in D-WARD, time are divided into a set of uniform intervals, called *observation period*, which serves as a unit time frame to compute traffic statistics. In each observation period, new traffic statistics are compared against past statistics derived from normal traffic. Network flows are classified according to the comparison results. Moreover, the statistics and comparison results are then passed to the throttling component which generates rate-limiting rules based on the behavior of the monitored network flows.

Fig. 5.1 shows a possible deployment of D-WARD. As depicted in the figure, D-WARD is a separate unit that acquires traffic from the default gateway and feeds the gateway with rate-limiting rules.

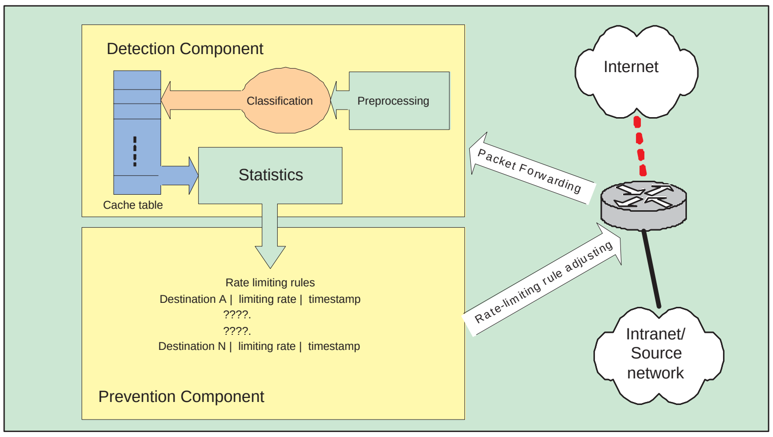


Figure 5.1: An example of the deployment of D-WARD

5.2.2 Attack Detection

In D-WARD, the aggregate traffic between monitored addresses and a correspondent host is defined as a *flow*. A flow is considered two-way if its data flow comprises packets originating from the sender and corresponding reply from the peer. TCP connections and several types of ICMP messages, such as "TimeStamp" and "echo", are typical examples of two-way flows. On the other hand, a flow is considered one-way (or uni-directional) if its data flow does not require reply messages in its normal operation. Traffics based on UDP protocol are examples of one-way flows.

For TCP flows, D-WARD defines a threshold that specifies the maximum allowed ratio of the number of packet sent to and received from in a flow. Notice that in the following context in this chapter, the ratio of the number of packet sent and received in a flow is refereed to as the *O/I* of the flow. Then, for TCP flows, whenever the O/I value of a flow breaches a pre-defined threshold, TCP_{rto}, the flow is classified as a DDoS attack flow. Similarly, for ICMP-based two-way flows, $ICMP_{rto}$ is used to define the maximum O/I value of an ICMP flow. In D-WARD's experimental settings, TCP_{rto} is set to 3 and $ICMP_{rto}$ is set to 1.1.

For a one-way flow, D-WARD defines three thresholds: an upper bound on the number of allowed monitored hosts issuing one-way connections to a correspondent host, a lower bound on the number of allowed packets in each one-way connection, and a maximum allowed sending rate. Whenever any of the three threshold is breached, the flow is considered attack. In D-WARD's experiment, the number of host in the same UDP flow must be smaller than 100. There must be at least one packet in each connection, and the maximum allowed sending rate is 10 MBps.

In D-WARD, a flow is classified as *normal*, *suspicious* or *attack* according to the comparison on the statistics derived from normal flows and the currently gathered statistics. If the statistics of a monitored flow does not consistent with normal model defined by thresholds mentioned above, the flow is classified as attack. If a flow that was ever classified as attack and the current comparison indicates compliance with normal flow model, it is classified as suspicious. Finally, if a flow is always compliant with normal flow model, it is classified as normal.

5.2.3 Attack Response

According to the comparison result passed from the observation component, the throttling component specifies allowed sending rates for monitored flows. D-WARD utilizes a flow control mechanism which is similar to the congestion control mechanism of TCP protocol. The sending rate is exponentially decreased in the first phase of attack response. Then, if further comparison indicates compliance with the normal flow model, a rate-limited flow can regain its bandwidth after the slow recovery and fast recovery process. On the other hand, a rate-limited flow can be more severely restrained if it does not comply with the rate limit and attempts to persistently rebel against the limited sending rate.

5.3 Proposed System

In this section, we first show that there are normal TCP flows with its O/I value which is greater than the threshold defined by D-WARD. This indicates that D-WARD would classify these TCP flows as attack while they are in their normal operations. This problem cannot be solved by using a sufficient large threshold since it will increase the false negative. Specifically, low rate attacks will not be detected. To cope with the problem, a new algorithm for detecting and limiting TCP-based DDoS attacks are presented herein.

It is worthy to notice that although DDoS attacks may take many different forms, it is reported [28,73,79] that over 94% of DDoS attacks use TCP. Thus, the scheme presented in this chapter may help defend against a majority of DDoS attacks. As to the detection of DDoS attacks based on of one-way flows, we suggest using the algorithm presented in D-WARD at current stage, but further enhancement is possible for the future work.

5.3.1 Basic Design Concepts

As mentioned above, D-WARD classifies a TCP flow as an attack flow if the O/I value of the flow is greater than TCP_{rto}. (Recall that, in D-WARD, this threshold is set to 3.) This approach suffers from the difficulty in determining an appropriate value for TCP_{rto}. It is because the O/I value of a TCP flow heavily depends on the implementation of TCP/IP protocol stack of the peers, and other factors such as round trip time and network congestion. This would result in a wide range of O/I values. For instance, Fig. 5.2 shows the average O/I values of TCP flows in a typical network consisting of 30

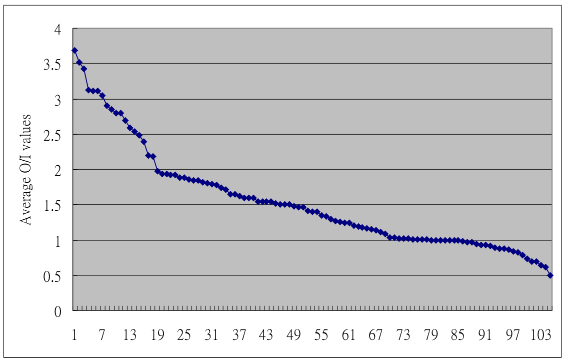


Figure 5.2: Average O/I values

personal computers. Operating systems installed in these computers include Windows 2000, Windows XP, FreeBSD, and Linux. As shown in the figure, there are flows with O/I values which are greater than 3. (The highest average O/I value is 3.68. It is observed in a flow consisting of only one FTP data connection.)

The observation motivates a new algorithm for detecting TCP-based DDoS attacks. The proposed algorithm exploits three essential characteristics of DDoS attacks, namely distribution, congestion and continuity, to detect the presence of DDoS attacks. First, distribution refers to the observation that DDoS attack scripts will normally infect as many insecure computer systems as possible so as to amplify the power of the DDoS attack. Therefore, in the monitored networks, if there is an increasing number of hosts attempting to send traffic to a destination host, a DDoS attack may just have been started. The statistics on the number of hosts sending packets to the same target will provide a valuable criterion for judging whether there is a DDoS attack or not. Second, DDoS attack usually lead to high packet loss rate toward the victim. Since monitoring packet loss rates of individual TCP flows would incur infeasible high cost, similar to D-WARD, the packet loss rate of a flow is abstractly represented as the O/I value of the flow. Third, continuity reflects to the observation that a DDoS attack usually lasts for an extended period of time. As we shall see later, this makes us be able to distinguish network congestion caused by DDoS attacks and other network events.

By taking advantage of the three DDoS characteristics, the proposed detection algorithm can classify TCP flows more precisely. In the proposed scheme, there are two phases: initialization phase and detection phase. In the initialization phase, the proposed scheme constructs initial profiles for TCP flows according to the past traffic in the flows.

In the profile database, each profile specifically represents the legitimate flow model of a TCP flow. Then, in the detection phase, traffic statistics are then compared with profiles. Profiles are dynamically adjusted to reflect the current behavior of monitored flows. In this way, different thresholds can be used to classify different TCP flows, and thus the efficiency of the detection algorithm can be effectively improved.

5.3.2 Detection Phase

In the proposed scheme, characteristics and thresholds of a flow are derived from the past traffic of the monitored flow. For each TCP flow, its traffic statistics computed from the current observation period are compared against the legitimate flow model defined by the profile of the flow so as to determine whether it is malicious or not. To better understand the proposed legitimate flow model, some notations are introduced as follows.

First, a two-way flow f is a collection of connections, and each connection is associated with a pair of IP addresses – an IP address in the set of monitored addresses and an IP address of the correspondent hosts. The former is referred to as *initial address* and the latter is *terminal addresses*. The number of distinct initial addresses in a flow f is denoted as S_f. For a connection c, n_c denotes the ratio of the number of packets originated from the initial address and received from the terminal address in one observation period in connection c. Then, n_f represents the average of the O/I value of all connections in flow f.

Furthermore, there are two threshold values, N_f and T_f, which help determine the malicious level of a monitored flow. N_f represents the mini threshold of a flow f. If $n_f \leq N_f$, then f is considered as a normal flow. T_f denotes the maximum allowed n_f. If $n_f \geq T_f$, then f is classified as an attack flow. If $N_f \leq n_f \leq T_f$, then further traffic statistics must be examined to determine the malicious level of the flow.

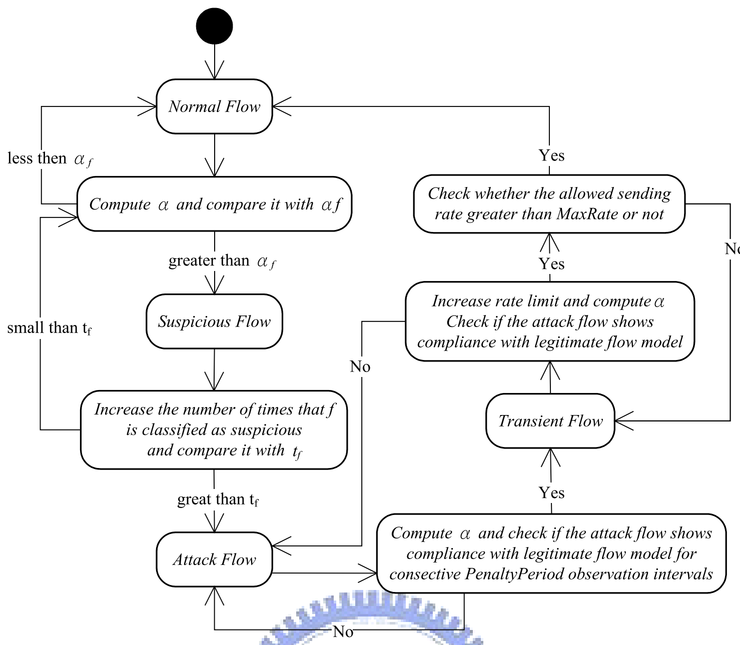
Then, the level of congestion and distribution can be quantified. Consider a flow f with $N_f \leq n_f \leq T_f$, the level of congestion of f refers to $(n_f - N_f)/(T_f - N_f)$. In this expression, we can clearly see that if the packet loss rate of the flow approaches T_f, the value of the expression will approach 1. On the other hand, if n_f approaches N_f, the value will approach 0. Next, the level of distribution is quantified as S_f/C, where C denotes a configuration parameter obtained from the past behavior of the monitored network (We will describe how to obtain this parameter later). Then, the level of congestion and distribution are combined and used to generate a value representing the malicious level of a monitored flow. Herein, the malicious level is denoted α and computed as follows. (In Eq. 5.1, λ is a number between 0 and 1, that is, $0 < \lambda < 1$. It is used to restrict the saturation of α between 0 and 1.)

$$\alpha = \frac{1-\lambda}{\lambda} * \sum_{i=1}^{\lfloor S_f/C \rfloor} (\lambda * \frac{n_f - N_f}{T_f - N_f})^i \tag{5.1}$$

It is worthy to note that α has two important characteristics. First, it is clear that α increases as n_f increases. In other words, if the packet loss rate of a monitored flow f gets higher, n_f will increase and consequently α increases. Second, α increases along with S_f even if n_f remains the same. This feature is especially useful in detecting DDoS attacks launched by attack programs which spoof source IP addresses. The α value will close to 0 when the monitored flow is in its normal operation. On the other hand, it will increase significantly when both the level of congestion and the level of distribution increases.

Although a surge of α value may indicate an DDoS attack that results in an abnormal increase in the packet loss rate or in the number of initial addresses in a flow, the α value can also go up due to regular network congestion. Nevertheless, the period of time the α value arises becomes a significant difference between the two causes. That is, normal network applications will stop sending more packets to a highly congested destination host after several attempts while DDoS attack scripts continually flush the victim for an extended period of time. With this observation, we can effectively distinguish a DDoS attack from a conventional network congestion by examining the length of time that DDoS attack signal lasts. This concept is implemented as follows. Consider a TCP flow f. α_f is a threshold that represents the maximum allowed α derived from the current network traffic. Once the threshold α_f is breached consecutively for t_f observation periods, f is considered a DDoS attack flow.

According to the proposed DDoS detection strategy, a network flow f can be classified into four types: *normal*, *suspicious*, *attack*, and *transient*. The transition of these types are depicted in Fig. 5.3. In brief, f is classified as a suspicious flow if $\alpha \geq \alpha_f$, where α is derived from the traffic in the current observation period. If α_f is breached for consecutive t_f observation periods, f is classified as an attack flow, and rate limiting techniques are applied to f. Once the traffic statistics of f shows compliance with legitimate flow model, i.e. $\alpha \leq \alpha_f$, for consecutive *PenaltyPeriod* observation periods, f is then classified as transient. For transient flows, rate limiting rules are carefully removed. When the allowed bandwidth of f reaches *MaxRate*, f is classified as a normal flow. Algorithm 10 shows pseudo code of the proposed detection algorithm.



Normal Flow
less then α_f
Compute α and compare it with αf
greater than α_f
Suspicious Flow
small than t_f
Increase the number of times that f is classified as suspicious and compare it with t_f
great than t_f
Attack Flow
No
Yes
Check whether the allowed sending rate greater than MaxRate or not
No
Yes
Increase rate limit and compute α Check if the attack flow shows compliance with legitimate flow model
No
Transient Flow
Yes
Compute α and check if the attack flow shows compliance with legitimate flow model for consective PenaltyPeriod observation intervals
No

Figure 5.3: Classification of Traffic Flow

In addition to the determination of the malicious level of monitored flows, it is desirable to update the thresholds for classifying network flows. This allows our scheme to learn the changing behavior of normal traffic, and dynamically adjust the thresholds according the current traffic statistics of monitored flows. For the adjustment of thresholds, attack traffic will be filtered out, and only traffic of a normal flow will be used to update thresholds. In this way, thresholds will not be polluted by attack traffic.

Updating a threshold is accomplished by feeding back the current traffic statistics. As we shall see shortly in this section, given a volume of historical traffic of the monitored flow f, we can derive T_f, and N_f for the monitored flow. In fact, by the same procedure, we can compute traffic statistics for each observation period. Assume that flow f is classified as normal in period i. Let $\widehat{T}_{f,i}$ denote the maximum allowed O/I value derived from the traffic volume in observation period i, and $T_{f,i}$ denote the same threshold used to classify flow f in period i (i.e., T_f of period i). Then, $T_{f,i+1}$ of the next observation period $(i + 1)$ can be computed in the same way:

$$T_{f,i+1} = \beta * \widehat{T}_{f,i} + (1-\beta) * T_{f,i}$$

In the similar fashion, N_f can be updated as follows.

$$N_{f,i+1} = \beta * \widehat{N}_{f,i} + (1-\beta) * N_{f,i}$$

where β is a configurable parameter ranging from zero to one. At present, we suggest β to be 1/2. However, its best value for a particular type of networks heavily relates to the variation of monitored network traffic, and may need further investigation.

5.3.3 Initialization Phase

It is clear that the settings of thresholds play an important role for the classification. As mentioned above, the learning phase is to compute these thresholds according to normal traffic of the monitored flows. It is worthy to note that the traffic used in the learning phase must be carefully examined and cannot contain attack traffic. Otherwise, the statistics derived will be incorrect and cannot be used to detect attacks. Due to this concern, Currently in the proposed scheme we perform off-line learning. That is, after all the thresholds are determined according to the historical traffic, the learning phase halts. This helps prevent our scheme from being polluted by on-line attack traffic. Next, the configurations of the threshold are described.

Consider parameters used in Eq. 5.1, i.e. T_f, N_f and C. Recall that T_f stands for the maximum allowed threshold of the O/I value of a monitored flow and N_f represents a mini threshold of the O/I value. Assume that the trail of historical network traffic is available which does not contain attacks. The traffic is partitioned into volumes in terms of observation periods, and then, thresholds are derived from the partitioned traffic volumes. In our scheme, the O/I value of each observation is measured. Then, we set $T_f = 2 * OI_{f,max}$ and $N_f = OI_{f,avg}$, where $OI_{f,max}$ denotes the maximum observed O/I value of flow f derived from historical traffic data and $OI_{f,avg}$ denotes the average O/I value. Next, Let C be the maximum number of distinct initial addresses in a flow during one observation period.

After T_f, N_f and C are configured, we can then compute a set of α values, one for each observation period. Then, we can set α_f to the average of the set of α values and

Algorithm 10 Detection Procedure

```
1: loop
2:   Collect IP packets received in one observation period.
3:   for each packet originating from monitored IP addresses do
4:     if Protocol = TCP then
5:       Classify the packet into a flow according to the destination IP address.
6:     end if
7:   end for
8:   calculate the O/I values of monitored TCP connections.
9:   for each flow (let the current flow be denoted as $f$) do
10:    if $N_f \leq n_f \leq T_f$ then
11:      compute $\alpha$ for flow $f$.
12:      if $\alpha \geq \alpha_f$ then
13:        increase the number of time that $f$ is classified as suspicious.
14:        if the number of times that $f$ is classified as suspicious $\geq t_f$ then
15:          generate a DDoS attack alert and classify the flow as attack.
16:          perform rate-limiting.
17:        end if
18:      else
19:        reset the number of times that $f$ is classified as suspicious.
20:      end if
21:    else if $n_f \geq T_f$ then
22:      Set $\alpha$ to 1, generate a DDoS attack alert and classify the flow as attack.
23:      perform rate-limiting
24:    end if
25:  end for
26: end loop
```

subsequently set t_f to be the maximum consecutive number of times that α_f is breached in the set.

5.3.4 Rate Limiting

In addition to detecting DDoS attacks, rate limiting is another component of the proposed scheme. In our approach, if a flow f is classified as attack, rate limiting technique will be applied to the flow in order to limit malicious traffic to a manageable level. One important design principle of our rate limiting strategy is that the rate limit applied to a malicious flow must reflect to current behavior of the flow. In this way, we can further restrict an ill-behaviored flow when it continually violates the legitimate flow model. From this point of view, the α value, which represents the malicious level of the monitored flow, serves as a rate limiting parameter. For the first time a flow is classified as an attack flow, the correspondent rate limit is:

$$rl = R * (1 - \alpha) \tag{5.2}$$

In Eq. 5.2, rl denotes the rate limit and R denotes the sending rate of the monitored for in the current observation interval. In the following observation periods, if the malicious flow does not show compliance to the legitimate flow model, it will be restrict further, according to the following formula:

$$rl_{new} = \min(rl_{old}, R) * (1 - \alpha) * \frac{P_s}{P_s + P_{drop}} \tag{5.3}$$

In Eq. 5.3, rl_{new} denotes a new rate limit to be applied on the malicious flow. rl_{old} denotes the rate-limit applied on the flow in previous observation interval. R represents the realized sending rate in previous observation interval. P_s is the total number of packets sent in the flow and P_{drop} is the total number of packets dropped because of the imposed rate limit.

In this way, flows that are part of DDoS attacks would be quickly restricted to a very low rate since the attack scripts would persistently send attack packets to the victim. Consequently, the fraction $(P_s)/(P_s + P_{drop})$ would become very low quickly.

Next, consider the case that the rate limited flow is mis-classified. In this case, TCP-based network applications will stop sending packets when the network is highly congested. Since the TCP/IP protocol will actively slow down the sending rate, the flow will show compliance with the legitimate flow model. In our approach, whenever an attack flow is compliant with the normal flow model for consecutive *PenaltyPeriod* observation periods, the flow is considered a transient flow and the recovery process begins. In the recovery process, rate limit are carefully removed according to the following equation:

$$rl_{new} = rl_{old} * \frac{1}{\alpha} * \frac{P_s}{P_s + P_{drop}} \tag{5.4}$$

In Eq. 5.4, it is clear that the speed of recovery is controlled by α and $P_s/(P_s + P_{drop})$. Both reflect the current behavior of the monitored flow. When the rate limit reaches *MaxRate*, a transient flow is classified as a normal flow, and rate limit is completely removed.

5.4 Performance Evaluation

To evaluate the performance of the proposed scheme, we implemented both prototypes of D-WARD and our approach on a machine which runs the FreeBSD operating system. In our experiment, two types of DDoS attacks are conducted: TCP SYN flooding attack and link overloading attack. In the TCP SYN flooding attack, each attack agent floods the victim with TCP SYN packet at the maximum rate of 100 KBps. In this experiment, we will show that attacks detected by D-WARD can also be detected by our approach. Even further, our scheme can detect the attacks earlier than D-WARD. Next, In the link overloading attack, each agents sends the victim at the maximum rate of 100KBps. The link bandwidth of the victim is restricted to 500KBps. This is accomplished by using Dummynet [95]. (there are in total 5 agents) In this experiment, we will show that our approach can detect attacks which cannot detected by D-WARD. For both types of attacks, we replicate the four attack scenarios tested in D-WARD. That is, constant rate attack, pulsing attack, increasing rate attack and gradual pulsing attack.

5.4.1 Experimental Results

Fig. 5.4, 5.5, 5.6, and 5.7 show the experimental results of TCP SYN attack. The x-axis denotes time measured in second and the y-axis stands for attack bandwidth measured in KB per second. The line with "x" symbols represents the attack bandwidth generated by attack agents. The line with triangle symbols represents attack bandwidth going through D-WARD, and the line with square symbols denotes the attack bandwidth passing by the proposed scheme. According to the figure, our scheme can detect the attack earlier than D-WARD. In other words, the defense system is able to respond to attack faster. This makes our scheme more effective than D-WARD mainly because the thresholds used in our scheme are continually adjusted and derived from the past behavior of the monitored flows.

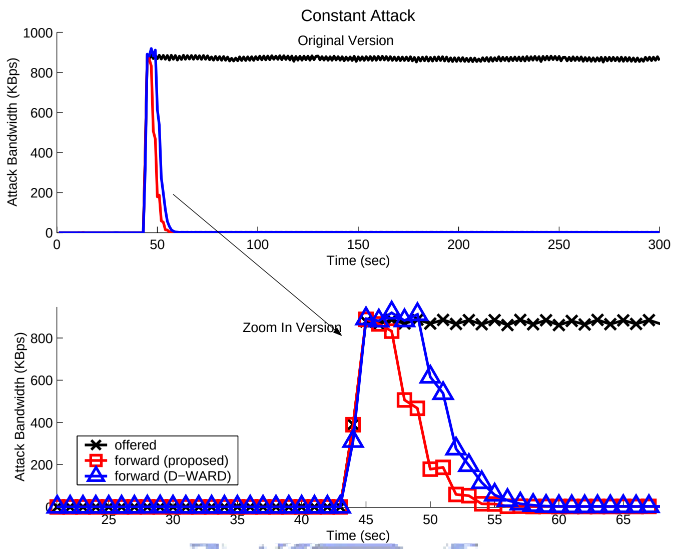


Figure 5.4: Constant SYNC attack.

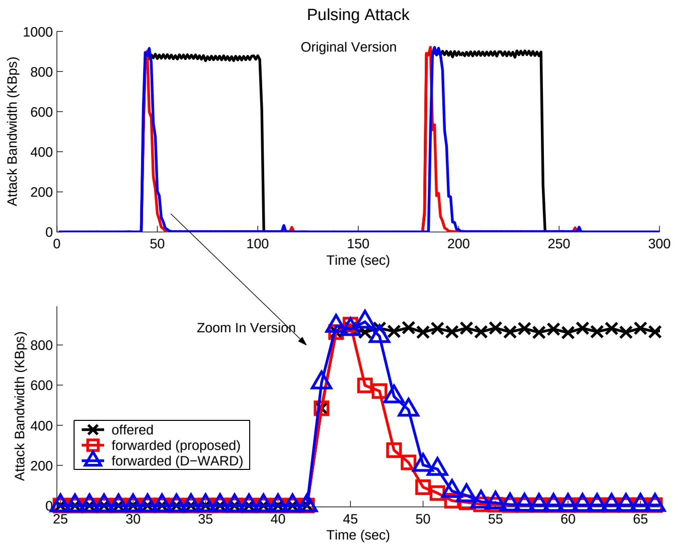


Figure 5.5: Pulsing SYNC attack.

Next, we examine the experimental results of link overloading attacks. In this ex-

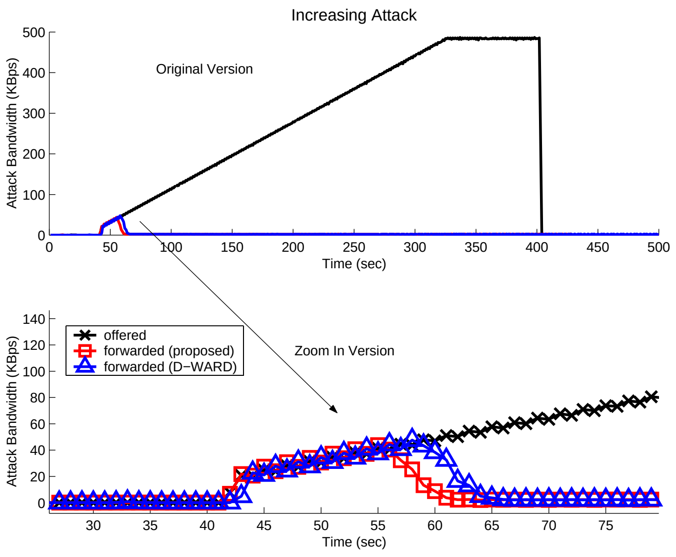


Figure 5.6: Increasing SYNC attack.

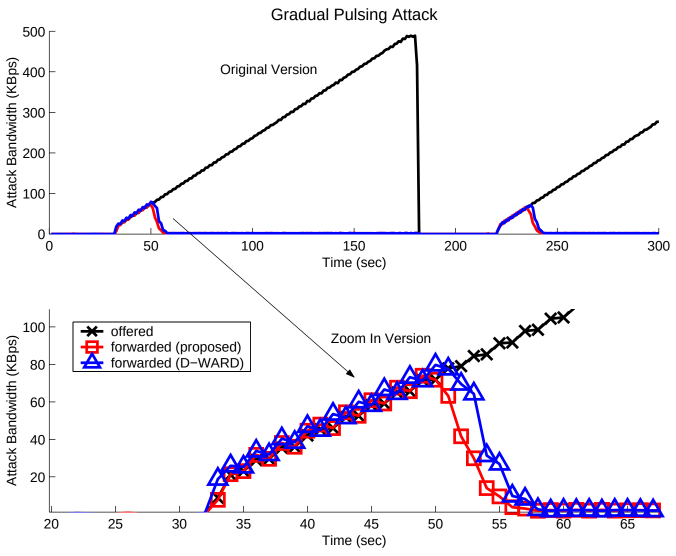


Figure 5.7: Gradual SYNC attack.

periment, by controlling the attack sending rate, the O/I value of the attack flow only reaches 2, smaller than threshold value 3 used in D-WARD. Thus, D-WARD is unable to detect the presence of the attack. On the other hand, the proposed scheme can identify the attack and perform subsequent rate limiting. Fig. 5.8, 5.9, 5.10, and 5.11 show the experimental result. Similarly, the x-axis denotes time measured in second and the y-axis stands for attack bandwidth measured in KB per second. The line with "x" symbols represents the attack bandwidth generated by attack agents. The line with triangle symbols represents attack bandwidth passing by D-WARD, and the line with square symbols denotes the attack bandwidth going through the proposed scheme. The experimental results show that our scheme is capable of detecting attacks with a O/I value which is smaller than 2, while D-WARD cannot. According to previous analysis and experiments, we can see that the proposed scheme has lower fast negatives and shorter response time. And, as mentioned previously, D-WARD will mis-classify network flows with a O/I value greater than 3 as attack flow. This contributes a major source of false postives for D-WARD. However, in our scheme, instead of using static threshold, criteria for identifying attack flows are derived from historical traffic of a monitored flow, and thus even the O/I of a monitored flow is greater than 3, our scheme will not classify it as an attack flow. In other words, with the ability of dynamically learning and updating thresholds, our scheme has lower false postives than D-WARD does.

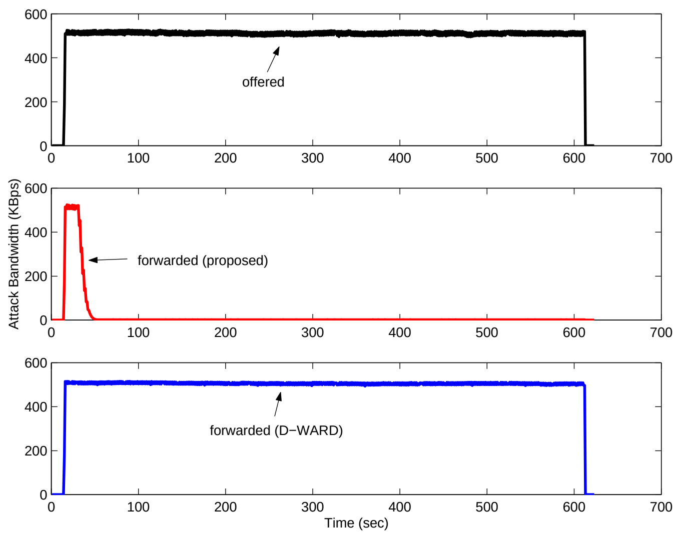


Figure 5.8: Constant bandwidth overloading attack.

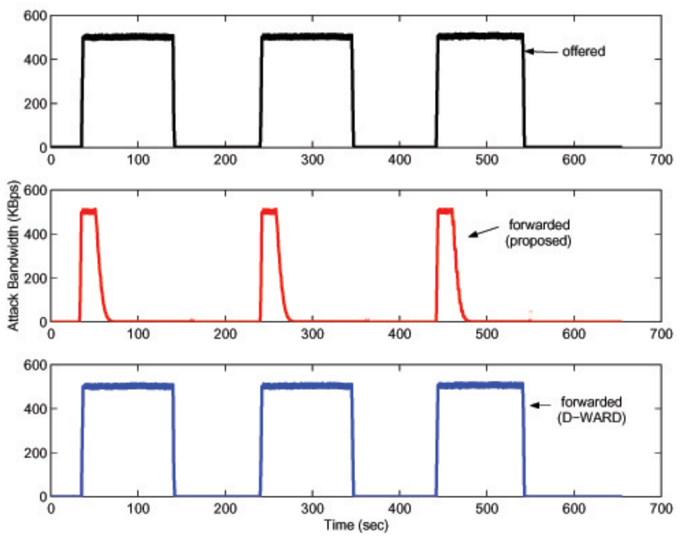


Figure 5.9: Pulsing bandwidth overloading attack.

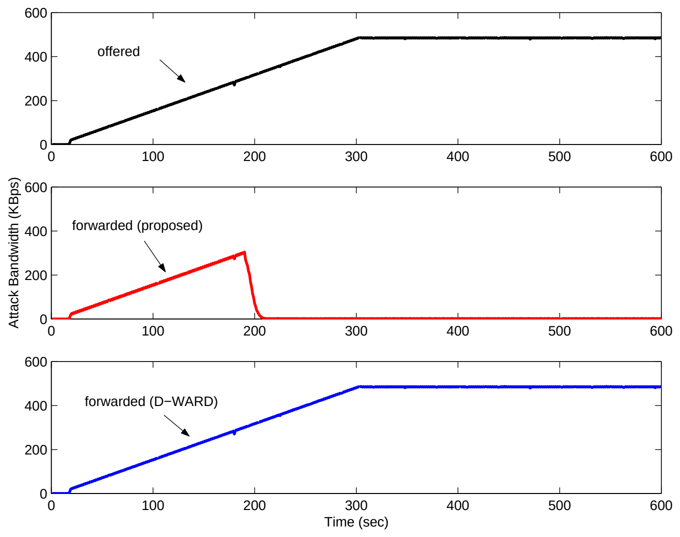


Figure 5.10: Increasing bandwidth overloading attack.

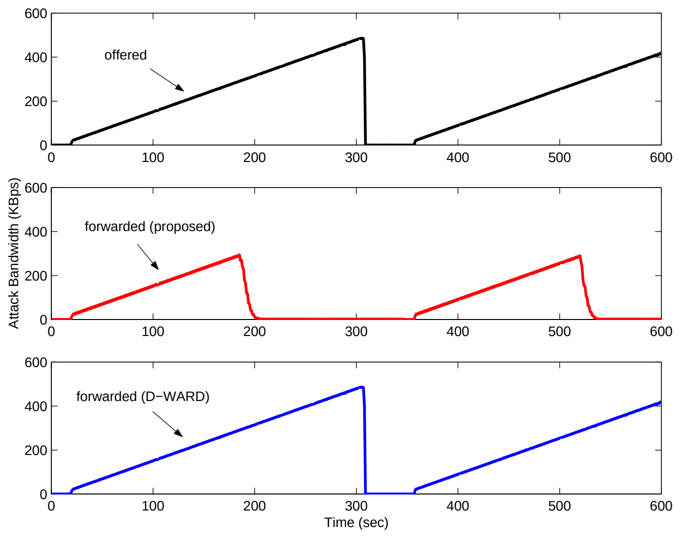


Figure 5.11: Gradual bandwidth overloading attack.

5.5 Summary

Technology resisting DDoS attacks has drawn considerable attention in recent years. However, most existing approaches suffer from either low detection rate, high deployment cost, or lack of effective attack response mechanisms. In this chapter, we presented a DDoS defense approach which monitors two-way traffic between a set of monitored IP addresses and the rest of the Internet. Our approach can accurately identify DDoS attack flows and consequently apply rate-limiting to the malicious network flows. In this way, DDoS attack traffic can be contained in source networks, and consequently lower the effectiveness of the attack. To effectively stop DDoS attacks, our approach needs to be deployed in routers serving as default gateways. With cooperative routers, our approach provides an effective defense mechanism against DDoS attacks.

Although the scheme presented in this chapter can effectively detect DDoS attacks based on two-way flows, several important issues need further investigation. For instance, one pressing problem not addressed in this chapter is how to establish the profile of a new type of flow that did not appear in historical traffic data. As mentioned previously, historical traffic used in the learning phase must not have attack traffic; otherwise, characteristics of normal flow behavior may not be derived. To achieve this, the simplest way is to manually examine the collected traffic before it can be passed to learning process. However, it is clear that this approach is not efficient since it requires an extensive amount of time to examine the traffic manually. Additionally, an effective profile management system is important and critical to the overall performance of the DDoS defense system. With all the systems putting together, the source-end DDoS defense can be quite effective and consequently deter DDoS attacks.

Chapter 6

Conclusions and Future Work

In this dissertation, we performed an intensive study on the problem of DDoS attacks and presented three defense approaches, each of which addressed DDoS attacks from a different direction. As shown in Chapter 2, the problem of DDoS attacks are unlikely solved by a single DDoS countermeasure. Instead, a suite of solutions for increasing the resistance of the Internet to DDoS attacks is desirable. This chapter first reviews characteristics of what we believe would be appropriate solutions against DDoS attacks, and then we summarize the propose defense approaches and discuss future directions which are worthy of further investigations.

We believe that an appropriate suite of DDoS solutions has the following characteristics.

- **Accurate detection and effective response:** Detection components that can accurately discriminate legal packets from attack ones and response components that can effectively block attack flows without collateral damages to legitimate packets are considered two basic build blocks of effective DDoS defense. This characteristic allows Internet servers to survive or to sustain their services to their legitimate clients when DDoS attacks occur.

- **Support for attack accountability:** The ability to locate the origins of attack streams is considered the first step toward the discovery of DDoS attackers. With the support for attack traceback, we may have a chance to deter malicious users from conducting DDoS attacks.

- **Feasibility of implementation and deployment:** Implementations of DDoS countermeasures should be feasible on the current Internet, e.g., should not impose substantial load on Internet routers, or require tight cooperations among different ISP networks. Enhancements on routers need to be minimized, and retrofits on routers must support incremental deployment. Retrofitting in a all-or-nothing fashion should be avoided.

The above-mentioned appropriate DDoS solutions appear challenging to obtain, and will certainly require not only enhancements on end host systems, but also software/firmware updates on Internet routers. With these required characteristics in mind, we will move on to summarize our work in defending against DDoS attacks, and see how close the three defense schemes presented in this dissertation approach an appropriate suite of solutions for addressing DDoS attacks.

6.1 Concluding Remarks

The first scheme, ANTID presented in Chapter 3, enables a victim to recognize spoofed packets and subsequently discard them on a per packet basis. ANTID only requires Internet routers to perform one hash operation in addition to normal processing of IP packets. An Internet server, which employs ANTID, needs to maintain for each of its communicating clients the mapping from the client's IP address to the corresponding path fingerprint. The construction and renewal of these mappings are performed in an on-demand fashion that helps reduce the cost of maintenance and storage requirement. In light of our performance analysis, ANTID can identify up to 99.95% spoofed IP packets and can discard these attack packets with little collateral damage to legitimate clients.

The second approach, EBRR, which supports the accountability of DDoS attacks is presented in Chapter 4. Specifically, EBRR addressed the problem of tracing individual attack packets. With the proposed path encoding technique, an Internet path can be encoded using a few bits. Given the limitation on available bits in IP packet header fields, an entire route of a packet is divided into a set of partial paths, which are then recorded and delivered to the destination host along with the packet itself or by a few ICMP packets. Our analysis indicated that the transmission overhead in EBRR is relatively low, and experimental results showed that in a network environment with 100 Mbps network links, the accuracy of identifying attack sources is up to 99.6%. For each Internet router, EBRR only requires one multiplication and one addition operation to encode path information, and normally this will neither disturb processing of IP packets nor require significant changes on the hardware of routers.

The third scheme, presented in Chapter 5 is a source-end approach whose goal is to detect ongoing DDoS attacks and to confine attack flows at their sources. Our approach inherited some DDoS detecting principles and system architecture from D-WARD and aimed at complementing rather than replacing D-WARD. In our scheme, a new DDoS detection technique is proposed to monitor two-way flows. Three essential features of DDoS attacks, i.e. distribution, congestion and continuity are integrated as a criteria for evaluating the malicious level of monitored network flows. Experimental results showed that the proposed scheme can shorten the time needed for detecting DDoS attacks and reduces both false positives and false negatives.

Finally we would like to remind readers that DDoS attacks is a complicated problem and an effective DDoS defense may comprise several defensive componments, each of which solves the DDoS attacks problem from a specific prespective. These defense systems are not necessary to be deployed at the same network elements, and they must be able to coexist without conflicts. Solutions presented in this dissertation addressed the DDoS problem from different points of view and can coexist without disturbing each other. However, it is also worthy to note that each solution has its own limitations. The victim-end defense focuses on identifying spoofed packets, and therefore it is weak in defending against DDoS attacks consisting of attack packets with real source IP addresses. The traceback scheme presented in this paper allows locating a participating Internet router that is nearest to the attack source. However, it cannot identify end hosts which inject attack packets into networks. Both victim-end and traceback schemes heavily rely on a basic assumption that Internet routers are secure against compromising. So far, to the author's knowledge, most victim-end and traceback schemes cannot survive with the existence of compromised routers. Once Internet routers are compromised, attacker can instruct them to erase any marks or path fingerprint on IP packets, and consequenlty defense systems counting these marks would not work correctly. The source-end defense system described in this dissertation complements the shortcomings of D-WARD system, rather than replacing it. Moreover, the proposed scheme only address DDoS attacks which are based on TCP packets. Techniques for detecting attacks with ICMP packets or UDP packets requires further much more investigations.

6.2 Future Work

Based the research results presented in this dissertation, some issues worth further investigation are listed in the following.

- **A plan for quick and easy deployment of DDoS defense:** Most DDoS solutions known at the time of writing of this dissertation involve some level of modifications on Internet servers, clients, switches and Internet routers. Usually required modifications need retrofitting and thus incurs financial costs. Ways for effectively deploying these defense approaches are seldom discussed. It is worthy to figure out a plan for integrating existing DDoS solutions and for quick and easy deployment of them.
- **Techniques for tracing actual DDoS attackers behind masters and zombies:** So far, most existing traceback mechanisms can only follow attack paths to attack zombies. Approaches to identify master machines which forward attack instructions and actual attackers who launch a DDoS attack need to be explored.
- **Traceback techniques in a hybrid network environment:** A study on the effectiveness of existing traceback approaches in the next generation Internet (NGI), such as IPv6 networks or networks with Network Address Translator (NAT) or mobile nodes, is needed. Approaches that can identify attackers in NGI need be developed.

DDoS attacks have long been regarded as a very hard problem, without one-size-fit-all solutions. Researchers addressed this problem by inventing different kinds of defensive approaches, and some of these solutions may form a complex solution for better solving the DDoS attack problem. In this dissertation, we have analyzed the problem of DDoS attacks and examined major trends in DDoS defense. We also presented three schemes for addressing DDoS attacks and experimental results indicated that the performance of proposed scheme are acceptable. Hopefully, the research results presented in this dissertation will serve as a useful foundation for future study in the field of DDoS defense and can inspire other researchers to develop more solutions to counter DDoS attacks.

Appendix A

Derviation of $F(L)$

To compute $F(L)$, we first consider a general case that the entire route is divided into k segments. The sequence of lengths of those segments are denoted as $(S_1^{L,k}, S_2^{L,k}, \ldots, S_k^{L,k})$. There are two representations of such a length sequence, as follows.

1. First, routers in the first segment are EBRR-enabled routers. That is, there are $S_1^{L,k}$ EBRR-enabled routers in the first segment and then followed by $S_2^{L,k}$ non-EBRR-enabled routers in the second segment. Next, in the third segment, there are $S_3^{L,k}$ EBRR-enabled routers. The segment of EBRR-enabled and non-EBRR-enabled routers are interchanged in the following segments.

2. Second, router in the first segment are non-EBRR-enabled routers and then followed by $S_2^{L,k}$ EBRR-enabled routers in the second segment. Similar to the first case, The segment of EBRR-enabled and non-EBRR-enabled routers are also interchanged in the following segments.

Consider the first case, the expected number of partial paths is:

$$1/2 * q^{L-k} * (1-q)^{k-1} * \sum_{j=0}^{\lfloor k/2 \rfloor} f(S_{2*j+1}^{L,k})$$

Similarly, the expected number of partial paths of the second case is:

$$1/2 * q^{L-k} * (1-q)^{k-1} * \sum_{j=1}^{\lfloor k/2 \rfloor} f(S_{2*j}^{L,k})$$

Since $A_i^{L,k}$ denote an instance of such sequence of lengths and combining the above two cases, we can find that the expect number of partial path of an instance $A_i^{L,k}$ is:

$$1/2 * q^{L-k} * (1-q)^{k-1} * \psi(A_i^{L,k}))$$

Under the same L, k constraint, there are in total C_{k-1}^{L-1} such sequences. Thus, the expected number of partial path in the case that the route is divided into k segments can be represented as:

$$1/2 * q^{L-k} * (1-q)^{k-1} * (\sum_{i=1}^{C_{k-1}^{L-1}} \psi(A_i^{L,k}))$$

Finally, since there route can be divided into from 1 to L segments, $F(L)$ can thus be computed as:

$$F(L) = \sum_{k=1}^{L} 1/2 * q^{L-k} * (1-q)^{k-1} * (\sum_{i=1}^{C_{k-1}^{L-1}} \psi(A_i^{L,k}))$$

Bibliography

[1] M. Adler, "Tradeoffs in Probabilistic Packet Marking for IP Traceback," *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pp. 407–418, 2002.

[2] G. Ahn, K. Kim, and J. Jang, "MF (Minority First) Scheme for Defeating Distributed Denial of Service Attacks," *Proceedings of Eighth IEEE International Symposium on Computers and Communication*, pp. 1233–1238, June 2003.

[3] T. Al-Herbish, "Raw IP Networking FAQ," Nov. 1999. [Online]. Available: http://www.whitefang.com/rin/rawfaq.html

[4] T. Al-Herbish, "Secure UNIX Programming FAQ," May 1999. [Online]. Available: http://www.faqs.org/faqs/unix-faq/programmer/secure-programming/

[5] K. J. Argyraki and D. R. Cheriton, "Active Internet Traffic Filtering: Real-time Response to Denial of Service Attacks," *CoRR*, vol. cs.NI/0309054, 2003. [Online]. Available: http://arxiv.org/abs/cs.NI/0309054

[6] K. J. Argyraki and D. R. Cheriton, "Protecting Public-Access Sites Against Distributed Denial-of-Service Attacks," *CoRR*, vol. cs.NI/0403042, 2004. [Online]. Available: http://arxiv.org/abs/cs.NI/0403042

[7] T. Baba and S. Matsuda, "Tracing Network Attacks to Their Sources," *IEEE Internet Computing*, vol. 6, pp. 20–26, Mar. 2002.

[8] J. Barlow and W. Thrower, "TFN2K - An Analysis," Feb. 2000. [Online]. Available: http://security.royans.net/info/posts/bugtraq_ddos2.shtml

[9] A. Belenky and N. Ansari, "IP Traceback with Deterministic Packet Marking," *IEEE Communications Letters*, vol. 7, no. 2, pp. 162–164, Apr. 2003.

[10] A. Belenky and N. Ansari, "Tracing Multiple Attackers with Deterministic Packet Marking (DPM)," *Proceedings of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 49–52, Aug. 2003.

[11] M. Bellare, R. Canetti, and H. Krawczyk, "Keyed Hash Functions and Message Authentication," *Proceedings of Crypto'96, LNCS 1109*, pp. 1–15, 1996.

[12] S. Bellovin, M. Leech, and T. Taylor, "ICMP Traceback Messages," Feb. 2003, available at: http://www.ietf.org/internet-drafts/draft-ietf-itrace-04.txt.

[13] B. H. Bloom, "Space/Time Trade-Offs in Hash Coding with Allowable Errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, July 1970.

[14] R. L. Carter and M. E. Crovella, "Server Selection Using Dynamic Path Characterization in Wide-Area Networks," *Proceedings of the IEEE INFOCOM*, pp. 1014 – 1021, Apr. 1997.

[15] CERT Coordination Center, "CERT Advisory CA-1996-01 UDP Port Denial-of-Service Attack," Feb. 1996. [Online]. Available: http://www.cert.org/advisories/CA-1996-01.html

[16] CERT Coordination Center, "CERT Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks," Sept. 1996. [Online]. Available: http://www.cert.org/advisories/CA-1996-21.html

[17] CERT Coordination Center, "CERT Advisory CA-1997-28 IP Denial-of-Service Attacks," Dec. 1997. [Online]. Available: http://www.cert.org/advisories/CA-1997-28.html

[18] CERT Coordination Center, "Denial of Service Attacks," Oct. 1997. [Online]. Available: http://www.cert.org/tech_tips/denial_of_service.html

[19] CERT Coordination Center, "Intruder Detection Checklist," Oct. 1997. [Online]. Available: http://www.cert.org/tech_tips/intruder_detection_checklist.html

[20] CERT Coordination Center, "List of Security Tools," Oct. 1997. [Online]. Available: http://www.cert.org/tech_tips/security_tools.html

[21] CERT Coordination Center, "CERT Advisory CA-1998-01 Smurf IP Denial-of-Service Attack," Jan. 1998. [Online]. Available: http://www.cert.org/advisories/CA-1998-01.html

[22] CERT Coordination Center, "CERTR Advisory CA-1999-17 Denial-of-Service Tools," Dec. 1999. [Online]. Available: http://www.cert.org/advisories/CA-1999-17.html

[23] CERT Coordination Center, "CERTR Incident Note IN-99-07 Distributed Denial of Service Tools," Jan. 1999. [Online]. Available: http://www.cert.org/incident_notes/IN-99-07.html

[24] CERT Coordination Center, "Results of the Distributed-Systems Intruder Tools Workshop," Nov. 1999. [Online]. Available: http://www.cert.org/reports/dsit_workshop-final.html,http://www.cert.org/reports/dsit_workshop.pdf

[25] CERT Coordination Center, "CERT Incident Note IN-2000-05 mstream Distributed Denial of Service Tool," May 2000. [Online]. Available: http://www.cert.org/incident_notes/IN-2000-05.html

[26] CERT Coordination Center, "CERTR Advisory CA-2000-01 Denial-of-Service Developments," Jan. 2000. [Online]. Available: http://www.cert.org/advisories/CA-2000-01.html

[27] CERT/CC, "CERT Advisory CA-2000-21 Denial-of-Service Vulnerabilities in TCP/IP Stacks," http://www.cert.org/advisories/CA-2000-21.html.

[28] K. C. Chang, "Defending against Flooding-Based Distributed Denial-of-Service Attacks: A Tutorial," *IEEE Communications Magazine*, vol. 40, pp. 42–51, Oct. 2002.

[29] C.-M. Cheng, H. Kung, K.-S. Tan, and S. Bradner, "ANON: An IP-layer Anonymizing Infrastructure," *Proceedings of the third DARPA Information Survivability Conference and Exposition*, vol. 2, pp. 78–80, Apr. 2003.

[30] B. Cheswick, H. Burch, and S. Branigan, "Mapping and Visualizing the Internet," *Proceedings of USENIX Annual Technical Conference*. [Online]. Available: http://www.usenix.org/publications/library/proceedings/usenix2000/general/cheswick.html June 2000.

[31] M. C. Chuah, W. C. Lau, Y. Kim, and H. Chao, “Transient Performance of PacketScore for blocking DDoS attacks,” *IEEE International Conference on Communications*, vol. 4, pp. 1892–1896, June 2004.

[32] K. Claffy, T. E. Monk, and D. McRobb, “Internet tomography,” June 1999. [Online]. Available: http://www.nature.com/nature/webmatters/tomog/tomog.html

[33] D. Cook, W. Morein, A. Keromytis, V. Misra, and D. Rubenstein, “WebSOS: Protecting Web Servers From DDoS Attacks,” *The 11th IEEE International Conference on Networks*, pp. 461–466, Oct. 2003.

[34] T. Darmohray and R. Oliver, “Hot Spares For DoS Attacks,” *;login:*, July 2000. [Online]. Available: http://www.usenix.org/publications/login/2000-7/apropos.html

[35] David A. Wheeler, “Secure Programming for Linux and Unix HOWTO,” Mar. 2003. [Online]. Available: http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO.html

[36] David Dittrich and George Weaver and Sven Dietrich and Neil Long, “The mstream distributed denial of service attack tool,” May 2000. [Online]. Available: http://staff.washington.edu/dittrich/misc/mstream.analysis.txt

[37] A. Daviel, “Rkdet,” Aug. 2002. [Online]. Available: http://vancouver-webpages.com/rkdet/

[38] D. Dean, M. Franklin, and A. Stubblefield, “An Algebraic Approach to IP Traceback,” *ACM Transactions on Information and System Security*, vol. 5, no. 2, pp. 119–137, May 2002.

[39] R. Deraison, “Nessus Open Source Vulnerability Scanner Project.” [Online]. Available: http://www.nessus.org/

[40] T. Dierks and C. Allen, “The TLS protocol version 1.0,” Internet Engineering Task Force, RFC 2246, Jan. 1999. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2246.txt

[41] S. Dietrich and N. Long, “An analysis of the Shaft distributed denial of service tool,” Mar. 2000. [Online]. Available: http://security.royans.net/info/posts/bugtraq_ddos3.shtml

[42] S. Dietrich, N. Long, and D. Dittrich, “Analyzing Distributed Denial Of Service Tools: The Shaft Case,” *Proceedings of 14th Usenix Systems Administration Conference*, pp. 329–339. [Online]. Available: http://www.usenix.org/events/lisa2000/full_papers/dietrich/dietrich.pdf Dec. 2000.

[43] D. Dittrich, “The DoS Project’s trinoo distributed denial of service attack tool,” Oct. 1999. [Online]. Available: http://staff.washington.edu/dittrich/misc/trinoo.analysis

[44] D. Dittrich, “The stacheldraht distributed denial of service attack tool,” Dec. 1999. [Online]. Available: http://staff.washington.edu/dittrich/misc/stacheldraht.analysis

[45] D. Dittrich, “The Tribe Flood Network distributed denial of service attack tool,” Oct. 1999. [Online]. Available: http://staff.washington.edu/dittrich/misc/tfn.analysis

[46] Y. Fan, H. Hassanein, and P. Martin, “Proactively Defeating Distributed Denial of Service Attacks,” *IEEE Canadian Conference on Electrical and Computer Engineering*, vol. 2, pp. 1047–1050, May 2003.

[47] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, “DDoS Tolerant Networks,” *Proceedings of the third DARPA Information Survivability Conference and Exposition*, vol. 2, pp. 73–75, Apr. 2003.

[48] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, “Statistical Approaches to DDoS Attack Detection and Response,” *Proceedings of DARPA Information Survivability Conference and Exposition*, vol. 1, pp. 303–314, Apr. 2003.

[49] P. Ferguson and D. Senie, “Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing,” Internet Engineering Task Force, RFC 2827, May 2000. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2827.txt

[50] Fyodor, "Nmap: Network Mapper." [Online]. Available: http://www.insecure.org/nmap/index.html

[51] S. Gibson, "Distributed Reflection Denial of Service: Description and Analysis of a Potent, Increasingly Prevalent, and Worrisome Internet Attack," Feb. 2002. [Online]. Available: http://grc.com/dos/drdos.htm

[52] V. Gilgor, "A Note on the Denial-of-Service Problem," *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 139–149, 1983.

[53] M. T. Goodrich, "Efficient Packet Marking For Large-Scale IP Traceback," *Proceedings of the 9th ACM Conference on Computer and Communication Security*, pp. 117–126, 2002.

[54] B. Huffaker, M. Fomenkov, D. Moore, and E. Nemeth, "Measurements of the Internet Topology in the Asia-Pacific Region," *Proceedings of INET Conference.* [Online]. Available: http://www.caida.org/outreach/papers/asiapaper/ July 2000.

[55] Internet Security Systems, "Trinity v3 Distributed Denial of Service Tool," Sept. 2000. [Online]. Available: http://xforce.iss.net/xforce/alerts/id/advise59

[56] J. Ioannidis and S. M. Bellovin, "Implementing Pushback: Router-Based Defense Against DDoS Attacks," *Proceedings of Network and Distributed System Security Conference*, pp. 79–86. [Online]. Available: http://www.isoc.org/isoc/conferences/ndss/02/proceedings/papers/ioanni.pdf Feb. 2002.

[57] C. Jin, H. Wang, and K. G. Shin, "Hop-Count Filtering: An Effective Defense Against Spoofed DDoS Traffic," *Proceedings of ACM Conference on Computer and Communications Security*, pp. 30–41, Oct. 2003.

[58] J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites," *Proceedings of IEEE International World Wide Web Conference*, pp. 252–262, May 2002.

[59] M. Kenney, "Ping of Death," Oct. 1996. [Online]. Available: http://www.insecure.org/sploits/ping-o-death.html

[60] A. D. Keromytis, V. Misra, and D. Rubenstein, "SOS: Secure Overlay Services," *Proceedings of the 2002 ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 61–72, Aug. 2002.

[61] A. D. Keromytis, V. Misra, and D. Rubenstein, "SOS: An Architecture for Mitigating DDoS Attacks," *IEEE Journal on Selected Areas in Communications*, vol. 22(1), pp. 176–188, Jan. 2004.

[62] S. Keshav, *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network.* Addison Wesley, 1997.

[63] Y. Kim, J.-Y. Jo, H. Chao, and F. Merat, "High-Speed Router Filter for Blocking TCP Flooding under DDoS Attack," *IEEE International Conference on Performance, Computing, and Communications*, pp. 183–190, Apr. 2003.

[64] Y. Kim, W. C. Lau, M. C. Chuah, and H. Chao, "Packetscore: Statistics-based Overload Control against Distributed Denial-of-Service Attacks," *Proceedings of IEEE INFOCOM*, vol. 4, pp. 2594–2604, Mar. 2004.

[65] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," Internet Engineering Task Force, RFC 2104, Feb. 1997. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2104.txt

[66] W. E. Kuhnhauser, "Root Kits: An Operating Systems Viewpoint," *ACM SIGOPS Operating System Review*, vol. 38, no. 1, pp. 12–23, 2004.

[67] H. T. Kung, S. Bradner, and K.-S. Tan, "An IP-Layer Anonymizing Infrastructure," *Proceedings of MILCOM*, vol. 1, pp. 389–394, Oct. 2002.

[68] H. T. Kung, C.-M. Cheng, K.-S. Tan, and B. S., "Design and Analysis of an IP-Layer Anonymizing Infrastructure," *Proceedings of the third DARPA Information Survivability Conference and Exposition*, vol. 1, pp. 62–75, Apr. 2003.

[69] H. Lee and K. Park, "On the Effectiveness of Probabilistic Packet Marking for IP Traceback under Denial of Service Attack," *Proceedings of IEEE INFOCOM Conference*, pp. 338–347, Apr. 2001.

[70] J. Li, J. Mirkovic, M. Wang, P. Reiher, and L. Zhang, "Save: Source Address Validity Enforcement Protocol," *Proceedings of IEEE INFOCOM*, vol. 3, pp. 1157–1566, June 2001.

[71] J. Li, M. Sung, J. Xu, and L. Li;, "Large-Scale IP Traceback in High-Speed Internet: Practical Techniques and Theoretical Foundation," *Proceedings of IEEE Symposium on Security and Privacy*, pp. 115–129, May 2004.

[72] L. Li and G. Lee, "DDoS Attack Detection and Wavelets," *Proceedings of International Conference on Computer Communications and Networks*, pp. 421–427, Oct. 2003.

[73] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling High Bandwidth Aggregates in the Network," *ACM Computer Communications Review*, vol. 32, no. 3, pp. 62–73, July 2002.

[74] S. Mann and E. L. Mitchell, *Linux System Security: The Administrator's Guide to Open Source Security Tools*. Prentice Hall PTR, 1999.

[75] D. L. Mills, "Network time protocol (NTP)," Internet Engineering Task Force, RFC 958, Sept. 1985. [Online]. Available: http://www.rfc-editor.org/rfc/rfc958.txt

[76] J. Mirkovic, G. Prier, and P. Reiher, "Attacking DDoS at the Source," *Proceedings of International Conference on Network Protocols*, pp. 312–321, Nov. 2002.

[77] J. C. Mogul and S. E. Deering, "Path MTU Discovery," Internet Engineering Task Force, RFC 1191, Nov. 1990. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1191.txt

[78] D. Moore, G. Voelker, and S. Savage, "Inferring Internet Denial of Service Activity," *Proceedings of USENIX Security Symposium*. [Online]. Available: http://www.usenix.org/events/sec01/moore.html Aug. 2001.

[79] D. Moore, G. Voelker, and S. Savage, "Inferring Internet Denial-of-Service Activity," *Proceedings of USENIX Security Symposium*, Aug. 2001.

[80] R. M. Needham, "Denial of Service: An Example," *Communications of the ACM*, vol. 37, no. 11, pp. 42–46, 1994.

[81] C. Papadopoulos, R. Lindell, A. Mehringer, J.and Hussain, and R. Govindan, “COSSACK: Coordinated Suppression of Simultaneous Attacks,” *Proceedings of the third DARPA Information Survivability Conference and Exposition*, vol. 1, pp. 2–13, Apr. 2003.

[82] K. Park and H. Lee, “On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets.” *Proceedings of SIGCOMM*, pp. 15–26, 2001.

[83] V. Paxson, “End-to-End Routing Behavior in the Internet,” *IEEE/ACM Transactions on Networking*, vol. 5(5), pp. 601–615, Oct. 1997.

[84] V. Paxson, “An analysis of Using Reflectors for Distributed Denial-of-Service Attacks,” *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 3, pp. 38–47, 2001.

[85] T. Peng, C. Leckie, and K. Ramamohanarao, “Protection from Distributed Denial of Service Attacks using History-based IP Filtering,” *Proceedings of IEEE International Conference on Communications*, vol. 1, pp. 482–486, May 2003.

[86] T. Peng, C. Leckie, and K. Ramamohanarao, “Adjusted Probabilistic Packet Marking for IP Traceback,” *Proceedings of the Second International IFIP-TC6 Networking Conference*, pp. 697–708, May 2002.

[87] T. Peng, C. Leckie, and K. Ramamohanarao, “Detecting Distributed Denial of Service Attacks Using Source IP Address Monitoring,” The University of Melbourne,Australia, Tech. Rep., 2002. [Online]. Available: http://www.ee.mu.oz.au/pgrad/taop/research/detection.pdf

[88] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, “Efficient and Secure Source Authentication for Multicast,” *Proceedings of Network and Distributed System Security (NDSS) Symposium*, pp. 35–46. [Online]. Available: http://www.isoc.org/isoc/conferences/ndss/01/2001/papers/perrig.pdf Feb. 2001.

[89] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, “The TESLA Broadcast Authentication Protocol,” *RSA Cryptobytes*, vol. 5, 2002.

[90] J. B. Postel, “Internet Protocol,” Internet Engineering Task Force, RFC 791, Sept. 1981. [Online]. Available: http://www.rfc-editor.org/rfc/rfc791.txt

[91] J. B. Postel, "Character Generator Protocol," Internet Engineering Task Force, RFC 864, May 1983. [Online]. Available: http://www.rfc-editor.org/rfc/rfc864.txt

[92] J. B. Postel, "Echo Protocol," Internet Engineering Task Force, RFC 862, May 1983. [Online]. Available: http://www.rfc-editor.org/rfc/rfc862.txt

[93] J. Postel, "Internet Control Message Protocol," Internet Engineering Task Force, RFC 792, Sept. 1981. [Online]. Available: http://www.rfc-editor.org/rfc/rfc792.txt

[94] E. Rescorla, "HTTP over TLS," Internet Engineering Task Force, RFC 2818, May 2000. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2818.txt

[95] L. Rizzo, "Dummynet: A Simple Approach to the Evaluation of Network Protocols," *ACM Computer Communication Review*, vol. 27, no. 1, pp. 31–41, Jan. 1997.

[96] L. A. Sanchez, W. C. Milliken, A. C. Snoeren, F. Tchakountio, C. E. Jones, S. T. Kent, C. Partridge, and W. T. Strayer, "Hardware Support for a Hash-Based IP Traceback," *Proceedings of the Second DARPA Information Survivability Conference*, pp. 146–152, June 2001.

[97] S. Savage, D. Wetherall, A. R. Karlin, and T. Anderson, "Practical Network Support for IP Traceback," *Proceedings of SIGCOMM Conference*, pp. 295–306, Aug. 2000.

[98] S. Savage, D. Wetherall, A. R. Karlin, and T. Anderson, "Network Support for IP Traceback," *IEEE/ACM Transactions on Networking*, vol. 9, no. 3, pp. 226–237, June 2001.

[99] B. Schneier, "Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)," *Proccedings of Fast Software Encryption, Cambridge Security Workshop*, pp. 191–204. Springer-Verlag, 1994.

[100] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer, "Hash-Based IP Traceback," *Proceedings of the ACM SIGCOMM Conference*, pp. 3–14, Aug. 2001.

[101] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, W. T. Strayer, and S. T. Kent, "Single-Packet IP Traceback," *IEEE/ACM Transactioins on Networking*, vol. 10, no. 6, pp. 721–734, 2002.

[102] D. Song and A. Perrig, "Advanced and Authenticated Marking Schemes for IP Traceback," *Proceedings of IEEE INFOCOM Conference*, pp. 878–886, Apr. 2001.

[103] M. Sung and J. X, "IP Traceback-Based Intelligent Packet Filtering: A Novel Technique for Defending against Internet DDoS Attacks," *Proceedings of International Conference on Network Protocols*, pp. 302–311, Nov. 2002.

[104] M. Sung and J. X, "IP Traceback-Based Intelligent Packet Filtering: A Novel Technique for Defending against Internet DDoS Attacks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, no. 9, pp. 861–872, Sept. 2003.

[105] W. Theilmann and K. Rothermel, "Dynamic Distance Maps of the Internet," *Proceedings of the IEEE INFOCOM*, vol. 1, pp. 275–284, Mar. 2000.

[106] R. Thomas, B. Mark, T. Johnson, and J. Croall, "NetBouncer: Client-legitimacy-based High-performance DDoS Filtering," *Proceedings of the third DARPA Information Survivability Conference and Exposition*, vol. 1, pp. 14–25, Apr. 2003.

[107] K. K. Wan and R. K. Chang, "Engineering of a Global Defense Infrastructure for DDoS Attacks," *Proceedings of 10th IEEE International Conference on Networks*, pp. 419–427, Aug. 2002.

[108] P. A. Watson, "Slipping in the Window: TCP Reset Attacks," *cansecwest/core conference*. [Online]. Available: http://cansecwest.com/csw04/csw04-Watson.doc 2004.

[109] J. M. Winett, "The Definition of a Socket," Internet Engineering Task Force, RFC 147, May 1971. [Online]. Available: http://www.rfc-editor.org/rfc/rfc147.txt

[110] D. Wreski, "Linux Security Administrator's Guide," 1998. [Online]. Available: http://www.nic.com/~dave/SecurityAdminGuide/SecurityAdminGuide.html

[111] Y. Xiang, Y. Lin, W. Lei, and S. Huang, "Detecting DDOS attack based on network self-similarity," *IEE Proceedings on Communications*, vol. 151, pp. 292–295, June 2004.

[112] J. Xu and W. Lee, "Sustaining Availability of Web Services under Distributed Denial of Service Attacks," *IEEE Transactions on Computers*, vol. 52, pp. 195–208, Feb. 2003.

[113] D. Xuan, S. Chellappan, X. Wang, and S. Wang, "Analyzing the Secure Overlay Services Architecture under Intelligent DDoS Attacks," *Proceedings of 24th International Conference on Distributed Computing Systems*, pp. 408–417, May 2004.

[114] A. Yaar, A. Perrig, and D. Song, "SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks," *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 130–143, May 2004.

[115] A. Yaar, A. Perrig, and D. Song, "Pi: A Path Identification Mechanism to Defend against DDoS Attacks," *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 93–109, May 2003.

Curriculum Vitae

Lee, Fu-Yuan was born in Taoyuan, Taiwan, in 1976. He received his B.S. degree in Computer Science and Information Engineering (CSIE) from National Chiao Tung University (NCTU) in 1998. He is currently a Ph.D. candidate in CSIE, NCTU. His research interests include computer networks, network security and system security.

Publicstion List

- **Journal papers**

  1. **<u>Fu-Yuan Lee</u>** and Shiuhpyng Shieh "Scalable and Lightweight Key Distribution for Secure Group Communications," *ACM/PH International Journal of Network Management*, volume 14, issue 4, pp. 167-176, May/June 2004.
  2. **<u>Fu-Yuan Lee</u>**, Shiuhpyng Shieh and Ya-Wen Lin, "Accelerating Network Services by Fast Packet Classification," *Computer Communications*, volume 27. number 16, pp. 1637-1646, 2004.
  3. Shu-Ming Cheng, Shiuhpyng Shieh, Wen-Her Yang, **<u>Fu-Yuan Lee</u>** and Jia-Ning Luo, "Design Authentication Protocols for Third Generation Mobile Communication Systems," *Journal of Information Science and Engineering*, volume 21, number 2, pp. 361-378, 2005.
  4. **<u>Fu-Yuan Lee</u>** and Shiuhpyng, "Defending against Spoofed DDoS Attacks with Path Fingerprint," accepted, *Computers & Security.*
  5. **<u>Fu-Yuan Lee</u>** and Shiuhpyng Shieh, "Packet Classification Using Diagonal-Based Tuple Space Search," in revision, *Computer Networks Journal.*
  6. **<u>Fu-Yuan Lee</u>**, Shiuhpyng Shieh and Ming-How Yang, "Edge-Based Record Route: A Novel Approach for Single Packet IP Traceback," in revision, *Journal of Computer Security.*

- **Conference papers**

  1. **<u>Fu-Yuan Lee</u>**, Shiuhpyng Shieh and Ming-How Yang, "Tracing the sources of Single-Packet Attacks," *Proceedings of Information Security Conference*, pp. 9-22, 2004.
  2. Kang-Hsien Chou, Shiuhpyng Shieh, Ming-How Yang and **<u>Fu-Yuan Lee</u>**, "Marking-Based Source Identification Scheme for Defending Against DDoS Attacks," *Proceedings of Information Security Conference*, pp. 23-38, 2004.
  3. Shiuhpyng Shieh, Ya-Win Lin, Zhi-Way Chen and **<u>Fu-Yuan Lee</u>**, "Fast Packet Classification for Improving the Performance of Secure Network Services," *Proceedings of Information Security Conference*, pp. 463-470, 2003.

4. **<u>Fu-Yuan Lee</u>**, Shiuhpyng Shieh, Jui-Ting Shieh and Sheng-Hsuan Wang, "Source-End Defense System against DDoS Attacks," *Proceedings of International Workshop on Advanced Development in Software and System Security*, Dec. 2003.

5. Shiuhpyng Shieh, **<u>Fu-Yuan Lee</u>** and Norton NG, "Packet-Based Load Balancing Scheme for Scalable Clustered IPsec Gateways," *Proceedings of Information Security Conference*, pp. 387-397, 2002.

6. **<u>Fu-Yuan Lee</u>**, Shiuhpyng Shihe and Lin-Yi Wu, "Addressing Sharing enabled Mobile IP utilizing NAPT," *Proceedings of International Computer Symposium*, pp. 731-740, 2002.

7. Shiuhpyng Shieh, Yung-Zen Lai, Ya-Ting Chang and **<u>Fu-Yuan Lee</u>**, "Clustered Architecture for High-Speed IPsec Gateways," *Proceedings of International Computer Symposium*, pp. 929-940, 2002.

- **Book Chapter**

1. **<u>Fu-Yuan Lee</u>**, Shiuhpyng Shieh, Jui-Ting Shieh and Sheng-Hsuan Wang, "Source-End Defense System against DDoS Attacks," Book Chapter, *Computer Security in the 21st Century*, Kluwer.

- **Patent**

1. Shiuhpyng Shieh, **<u>Fu-Yuan Lee</u>** and Yung-Zen Lai, "Method and System for Synchronous Packet Processing," *ROC patent no. I227082*, Jan. 2005.