ABSTRACT

CHIEN-LUNG WU. On Network-Layer Packet Traceback: Tracing Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) Attacks. (Under the direction of Dr. Shyhtsun Felix Wu and Dr. Arne A. Nilsson)

The objective of this research is to study the Internet Protocol (IP) traceback technique in defeating Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) attacks. Tracing attackers is the first and most important step to solve the DoS/DDoS problem. In this dissertation, two new traceback techniques, PHIL and Intention-Driven iTrace, are proposed and evaluated. Based on the IPSec infrastructure, previously, the Decentralized Source Identification for Network-based Intrusions (DECIDUOUS) module has been implemented and evaluated. However, in order to trace attack sources across different administrative domains securely, the notion of Packet Header Information List (PHIL) for IPSec is proposed to enhance DECIDUOUS module. Second, it is shown, in this thesis, that the iTrace (ICMP traceback, being standardized in IETF) has some serious drawbacks. To overcome these drawbacks, the Intention-Driven iTrace (ID-iTrace) and the Hybrid iTrace schemes are proposed. Our simulation results confirm that the original iTrace scheme is not able to handle low attack traffic well. From our simulation, the Hybrid iTrace scheme is evaluated and demonstrated to be an efficient and practical mechanism for tracing DoS/DDoS attacks.

On Network-Layer Packet Traceback: Tracing Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) Attacks

By

CHIEN-LUNG WU

A dissertation submitted to the Graduate Faculty of
North Carolina State University
In partial fulfillment for the
Requirements for the Degree of
Doctor of Philosophy

ELECTRICAL AND COMPUTER ENGINEERING

Raleigh

2003

APPROVED BY:

George N. Rouskas, Ph. D.

Edward Gehringer, Ph. D.

Shyhtsun Felix Wu, Ph. D.
Co-Chair of Advisory Committee

Arne A. Nilsson, Ph. D.
Chair of Advisory Committee

To my Lord,

to my parents,

to my wife Mei-Yu,

and our daughters, Grace and Praise.

PERSONAL BIOGRAPHY

CHIEN-LUNG WU was born in Taichung, Taiwan. He attended ChungYuang Christian University from 1983 to 1987, where he graduated with a B.S. degree in Electrical and Computer Engineering. After graduation, he entered Siemans Inc. in Taiwan as a hardware engineer and worked in R&D department for five years. Chien-Lung came to the United States of America in the fall of 1994 to study for his master's degree in Electrical and Computer Engineering at Syracuse University, N.Y. He obtained his M.S. degree in Electrical Engineering in 1995, and M.S. degree in Computer Engineering in 1996. In 1997, he continued his Ph.D. studies in the department of Electrical and Computer Engineering at North Carolina State University, Raleigh. While attending North Carolina State University, he was employed by Fujitsu Networks as a research assistant in mobileIP and Quality of Service Routing (QoSR) projects. From 1998 to 2001, he was a research assistant to Dr. Felix Wu, working on the DECIDUOUS and FNIISC projects. In 2002, he joined Delta Network as a Senior Software Engineer. His current research interests are attack-tracing mechanism design, network security and management, network routing protocol design, and embedded system design.

ACKNOWLEDGEMENTS

First, I would like to thank my advisor Dr. Shyhtsun Felix Wu. He taught me the essence of network security and the principles of research, and guided me in finishing this dissertation. I will never forget his encouragement and support.

I sincerely thank my committee members, Dr. Arne A. Nilsson, Dr. Edward Gehringer, and Dr. George N. Rouskas. Their valuable and constructive suggestions helped me to improve my dissertation. I am grateful for their time and efforts in assisting me during this process.

I would like to thank my colleagues: Xialiang Zhao, Zhu Fu, Ho-Yen Chang, Feiyi Wang and others. Many discussions, sharing, encouragement, and suggestions gave me great help in my studies.

I also want to thank my brothers and sisters from the Raleigh Chinese Christian Church. Their continuous support and prayers encouraged me to overcome a lot of frustrations in my life.

I would like to thank Dr. Albert Shih, Dr. Raymond Chen, and Dr. S. W. Kiu. They gave me many suggestions and advice, and encouraged me to finish this dissertation. Thank you for your support. A special word of thanks for Doris Kiu: I will never forget your help, teaching, and encouragement.

I know I owe my family a lot. My parents and parents-in-law have always given me their love and support all my life. My sisters and brothers have also given me all their support over the years. And my dear wife, Mei-Yu, always supports and continually prays for me. Over the years, her encouragement, patience, and love are always with me. I dedicate all my love and this dissertation to them.

TABLE OF CONTENTS

List of Figures

List of Tables

Chapter 1 Introduction

In today's internet, one of the thorniest problems in dealing with Denial-of-Service (DoS) and Distributed Denial-of Service (DDoS) attacks is to trace the attack sources, given the present state of the internet. There is an urgent need to thoroughly understand the problems involved and to build an efficient mechanism to address the tracing issues. This chapter presents the research motivation and outlines the overall structure of the whole project.

1.1 Problem definition and Research Motivation

1.1.1 DoS and DDoS attacks

A denial-of-service (DoS) attack [1] is an attack in which intruders intentionally take up much of the shared resources of a remote host or network, and as a result, legitimate users are denied from using network resources. The distributed denial of service (DDoS) [2, 3] attack is the distributed version of DoS attacks: the attackers compromised hundreds or thousands of machines and controlled those compromised machines to attack targets simultaneously.

Both DoS and DDoS attacks exploit the vulnerability of today's network infrastructure [4, 5, 6]. In the past several years, DoS/DDoS attacks have increased in frequency, severity, and sophistication [7, 8]. The 2002 CSI/FBI Computer Crime and Security Survey results [9] show that over 55% of respondents have been hit by DoS/DDoS attacks, which caused total losses of $18,370,500 in the first quarter of 2002. The widely known February 2000 DoS attacks [10], bringing down commercial websites such as Yahoo, eBay, Amazon, CNN, and ZDNet, are a painful reminder that DoS and DDoS attacks have become some of the severest network security problems.

1.1.2 Source accountability

The nature of IP protocol and IP routing only depends on the destination address but not the source IP address [11, 12] to route IP packets. Hence, the attackers can put any spoofed or forged IP address on the source address field of IP header to deceive intrusion detection systems (IDS) and victims. Therefore, a victim cannot count on using the information in the received attack packets to discover the identity of the original attack sources. This is the source accountability problem [13], which makes it very difficult to trace the attacker(s).

1.1.3 IP traceback problem

The IP traceback is a special mechanism that enables victims or intrusion detection systems (IDS) to trace attacks back to their origins and hopefully, stop attackers at the attack sources [6, 14, 15]. However, this is difficult to do because of the present state of source accountability (un-trusted source IP addresses). How to solve this IP traceback problem is one of the big challenges in network security research, especially in the area of eliminating DoS/DDoS attacks.

1.1.4 Research objective

Our objective is to design and develop a possible and practical traceback mechanism so that we can automatically trace the attack sources, narrow down the suspected area to a reasonable and small enough domain, and then stop the attacks right at the sources. In particular, we focused on DoS/DDoS attacks attempting to saturate network resources so as to prevent legitimate requests from being served.

1.2 IP Traceback Solutions: our proposals

In this dissertation, we developed and evaluated two new techniques to enhance the performance of IP traceback systems. The first approach is the IPSec-based source tracing scheme. As the IP security protocol (IPSec) [16] becomes a de facto standard, it is possible and convenient for us to develop a trace back mechanism by utilizing the IPSec framework. The second approach is the ICMP traceback scheme [17]. Using an efficient packet sampling mechanism, the intermediate routers can provide useful information for the victim to construct an attack route path. When victims or Intrusion Detection Systems obtain the information of the attack paths, we are able to trace the attackers and then stop the attacks immediately.

1.3 Research Contributions

The contributions of this thesis are:

- The design of Packet Header Information List (PHIL) and PHIL-APIs extends the utilization of the IPSec protocol and makes application-to-application security possible.
- The PHIL-switching technique provides a solution of tracing attack sources across different domains, which is a major issue of IPSec-based approach.

- The ICMP traceback (iTrace) [17] mechanism, one of the sampling-based approaches, is studied. Its weaknesses are investigated and simulated in our test bed.
- The Intention-Driven iTrace (ID-iTrace) [18] and Hybrid iTrace schemes are developed to enhance the original iTrace scheme at a practical level thus eliminating many of its weaknesses.

1.4 Dissertation Outline

This dissertation is divided into seven chapters. Chapter 1 introduces the IP traceback problem of DoS/DDoS attacks and the motivation of this research. Related works are provided in Chapter 2. Chapter 3 presents the IPSec-based approach. Chapter 4 introduces the ICMP traceback (iTrace) and presents our enhanced versions: the Intention-Driven iTrace (ID-iTrace) and the Hybrid iTrace schemes. A light-weigh authentication mechanism for iTrace/ID-iTrace is demonstrated in Chapter 5. Chapter 6 describes our simulation processes and shows our simulation results. Chapter 7 summarizes the contribution of this dissertation and discusses possible directions for future research.

Chapter 2 Related Works

In this chapter, a wide spectrum of research works [6, 17, 19, 20, 21, 22, 23, 24, 25] conducted in the area of IP traceback issues is reviewed and analyzed. An understanding of various approaches, including the ingress filter technique, attack-link identification (link-testing, control flooding, and IPSec-based tracing), logging-based scheme (logging and hash-based logging), and sampling-based scheme (probabilistic packet marking scheme and ICMP traceback scheme), provides us with background knowledge and also shows us the research directions in this field.

2.1 Ingress filtering

The ingress filtering approach [19] is proposed to prevent an attacker within an originating network from using forged IP addresses. This technique requires that a router has both sufficient power to examine the source address in every single packet and enough knowledge to distinguish between legitimate and illegitimate addresses. Hence, it is better to configure the ingress filtering in customer networks or at the border of internet service providers (ISP), where the address ownership is easy to maintain. Any packet with an IP address which cannot maintain the ownership or doesn't have the ownership will be dropped. Thus the ingress filtering approach greatly reduces forged IP problems and also minimizes the possibility of flooding targets with forged addresses.

However, there are several concerns about this scheme. First, the examination of every single packet will definitely degrade network performance. This is the major concern that ISPs do not want to use this functionality. Second, the effectiveness of ingress filtering technique depends on the widespread use of this method. If some ISPs do not cooperate to provide this feature, the source IP still cannot be trusted because it is unable to know whether the packet comes from filter-enabled or filter-disabled domains. Third, even though all ISPs have turned on the ingress filtering, DoS/DDoS attackers still are able to use forged IP addresses which are on the list of ISP's ownership.

2.2 Attack-link identification

This category of traceback schemes examines upstream links in order to identify attack flows. After each attack flow (from the victim's domain toward attack domains) has been confirmed, the attack path(s) can be discovered, and as a result, the attack sources can be located and then the attacks stopped. To examine the attack links, there are two proposed methods. One is to flood upstream links. If the attack decreases when one of the links is flooded, this link can be identified as an attack link. Link testing and controlled flooding [20] use this method to trace attack sources. Another approach which utilizes link authentication is the one used by IPSec-based tracing [21]: if attack packets come from an authenticated upstream link, this authenticated link is an attack link

2.2.1 Link testing

Link testing is a way of discovering the domain from which the attack packets actually originated by examining upstream links. However, this technique requires the attack flow to be continuous for a period of time. Once the link is identified, the victim will inform its upstream ISP. The upstream ISP will repeat this process by testing each of its upstream links and identifying which link is carrying the attack flow. This process will continue recursively until the attack flow is isolated to a stub network. Once this attack domain is located, an investigation can proceed internally to discover the true attacker(s).

This approach has a few problems. First, it relies on individuals to carry out the link testing manually. Second, if any ISP or domain between the true attacker and the victim refuses to cooperate or simply does not have the knowledge or manpower to cooperate, it is impossible to proceed beyond that point. Thus, this method relies on the fact that all ISPs between the attacker and victim are able and willing to cooperate.

2.2.2 Controlled flooding

One extension of link testing that can be performed automatically by the victim without the aid of intermediate ISP's is called controlled flooding [20]. The goal of controlled flooding is to recursively identify which upstream links are carrying the attack traffic by flooding all upstream links and analyzing the effect of this flooding on the attack flow. If flooding a certain link causes the attack flow to decrease, this link is probably being utilized by attackers. This method has an advantage over traditional link testing in that the victim can use it with no intervention on the part of intermediate ISPs. The fundamental flaw in this approach is that it requires flooding another domain's network in order to do the link testing. This, in itself, can be construed as a DoS attack.

2.2.3 IPSec-based IP traceback

Node-to-node authentication [26, 27], one of the functions of IPSec, provides an alternative way to identify attack links. Since IPSec [16] was widely deployed and became a de facto standard, it is convenient for us to build up a traceback mechanism on top of IPSec infrastructure. For example, if an authenticated tunnel from A to B is set up, one can be sure that any packet received by B from this tunnel is routed by node A. If someone applies this concept to trace attacks, then node A can be identified as the node which routes the attack packets. Therefore, if IPSec tunnels are set up between the victim and possible nodes, one is able to identify attack paths and ultimately locate attack sources. The Decentralized Source Identification for Network-based Intrusions (DECIDUOUS) project [21] utilized this idea to implement a traceback mechanism which was proved to be effective in locating attack sources.

Although the IPSec-based traceback mechanism can finally identify the attackers, there exist drawbacks in this approach. First, any IP traffic, including attack traffic and innocent traffic, would be pushed into IPSec tunnels for identifying attack flows. This method definitely affects and degrades network performance, especially in a high-speed network. In the case of flooding-style DDoS attacks, the situation becomes worse. Second, in general, the victim domain doesn't have any topology information of the entire Internet. Where and when to set up IPSec tunnels is a major concern. Last, if any transit routing domain cannot cooperate, this approach will not work.

On the other hand, if the suspect domains can be narrowed down to a small area (for instance, a university network domain), utilizing the authenticated tunnels to trace true attacks is still a valuable approach. Moreover, within a local administration domain, the network performance will not be greatly affected. Also the network topology within a local administration domain is known and as a result, someone can pre-setup IPSec tunnels to improve the identification performance. Furthermore, The PHIL (Packet Header Information List) and PHIL-switching techniques [28, 29], an extension of IPSec-tunnel mechanism, allow local administrators to set IPSec tunnels dynamically and automatically, thereby enhancing the tracing capability of the IPSec-based traceback mechanism. A detailed description of the IPSec-based traceback scheme and PHIL/PHIL-switch enhancement is presented in Chapter 3.

2.3 Logging-based IP traceback

Another category of IP Traceback solutions employs logging scheme at routers, which store information about forwarded packets. Later on, the victim of an attack can query a specific router to find out whether that router forwarded a specific packet. This approach is limited in practice due to the large storage requirement involved for any given router; this method also requires a long period of time and special schemes to query and analyze these logged data.

The hash-based IP traceback technique [30, 31] is a new log-based approach to address the space requirement, query overhead and susceptibility to DoS attack. Instead of storing the entire contents in the packet, which would require 20 to 1,500 bytes of storage per packet, only a 32-bit digest of specific packet-header fields is stored. Since this digest must be identical for a single packet traversing multiple routers, any fields in the packet header that may change from router to router (such as the TTL, checksum, type of service (TOS) and IP options) must be ignored. To compensate for the probability of a collision, a special data structure called Bloom filter [30] is utilized to store the digest information. However, instead of computing only a single digest, a Bloom filter computes k digests using k independent hash functions, each with a 32-bit uniform output. The output digests are used as indices of an array with a size of 2*32 bits, and the matching positions have all their bits set as 1 (initially, all positions are set as 0). Whether or not a certain packet has been forwarded by a particular router can be determined if all k positions in the Bloom filter are set as 1.

The problem with this approach, however, is that even under the above conditions it is possible to get a false positive from the data. Because the Bloom filter has limited space and each forwarded packet occupies k entries, the rate of false positives increases as more entries are marked with each forwarded packet. This limits the usefulness of the hash-based solution. Another disadvantage is the problems it creates for storage. If the Bloom filter which occupies approximately 512MB at every router is kept resident in memory, it will occupy space that routers normally reserve for storing received prefixes from BGP. An alternative will be to store the data in a secondary storage medium, such as a hard drive. The seek times and processor overhead to fetch data can significantly reduce the performance of the router.

2.4 Sampling-based IP traceback

This scheme, unlike other schemes which trace attacks in every single packet, only samples certain packets based on a randomly sampling scheme; these sampled packets carry important path information for the victim or the Intrusion Detection system to discover attack sources. Regarding the method of conveying path information, there are two categories of sampling-based approaches. One is the probabilistic packet marking [6, 32, 33] scheme which provides in-band path information by modifying IP header fields to encode path information. Another approach is the ICMP traceback [17] scheme which copies the sampled packet to generate out-of-band path information (ICMP packets).

Using the sampling scheme is a feasible solution to enhancing the efficiency of the traceback mechanism while minimizing the adverse effect on network performance. Given the current state of heavy network traffic, sampling-based scheme can be even more effective if we can optimize the sampling probability.

2.4.1 Probabilistic Packet Marking (PPM) scheme

This scheme asks each intermediate router to randomly sample packets in a probability ***p*** and then encode its identification (router ID) into these sampled packets. Each marked packet, like normal packets, will be forwarded to the destination address described in the IP header. After the destination host receives these marked packets, it can decode these packets to obtain each router's identification and figure out attack paths. For example, the destination host can examine suspect attack packets and discover possible attack paths and ultimately locate the true attackers.

There are several proposed approaches to implement the PPM scheme. The first one is called node sampling [6, 33], proposed by Savage in 2000. In this technique, an upstream router will randomly select packets with probability ***p*** and mark these packets by inserting its IP address in a special 32-bit field of IP header before forwarding this packet. Since each upstream router forwarding packets to the destination host has a probability of ***p*** to mark a packet, the router information could be overwritten by other routers [6]. Therefore, the probability for a victim to receive a marked packet from a router, which is *d* hops away from the victim, can be expressed as ($p* (1 - p)^{d-1}$) -- a function of distance *d* (in terms of hops). This probability will significantly decrease as the distance *d* increases. Thus to correctly construct the path at greater distances might be difficult. In situations while time is of the essence this will pose a serious problem. Furthermore, since this method just provide node identification, it is hard for the destination host to figure out the sequence of intermediate routers and then to discover correct routing paths.

To address this issue, Savage [6, 32] also proposed the edge sampling technique where an edge (a link between two routers) is stored into the marked packet along with the number of hops between the routers. This would require 72 bits of additional space to store two 32-bit IP addresses and one 8-bit distance field -- a significant increase when compared to node sampling. To solve the space problem, this technique uses an encoding scheme that allows the 72 bits of data to be encoded and then fragmented into several pieces. After the destination host receives enough marked packets, it is able to reassemble these pieces to obtain the edge identification.

Although the edge sampling approach solves the sequence and space problems, there are several drawbacks to this approach. First, it suffers from the extremely high process overhead required in marking packets. In addition, since the edge identifier is stored in the IP identification field of IP header, this mechanism would invalidate any fragmented IP packets. Furthermore, because the edge-id is fragmented to reduce space requirements, the edge sampling technique requires more marked packets to discover the attack paths. Another flaw in both edge and node sampling approaches is the lack of protection in cases when malicious users mark packets with false data.

The advanced marking scheme [22, 34] is proposed to address the overhead and authentication issues of edge sampling scheme. Instead of generating a 64-bit edge-id that later gets fragmented, this scheme simply uses two unique hash functions to compute the hash of one router IP address. The particular hash functions used in this scheme take 32-bit

IP addresses and return 11-bit hashes. These hashes are then XOR'd together to generate the edge-id. This edge-id is stored in the IP identification field (16 bits) as in the edge sampling approach, and the remaining 5 bits are used for the distance value (hop count). After receiving these marked packets, the destination host can compute the hash functions to identify nodes which marked these packets and use the distance value to decide the sequence of nodes of an attack path.

The Advanced Marking Scheme still does not provide protection from attempts to confuse path reconstruction by overwriting legitimate markings with garbage or false markings. Since all routers and the victim's path reconstruction tool use the same set of hash functions, these functions are easily accessible to attackers. An attacker can use the same hash functions to generate false markings to interfere with the process of path discovery. To solve the authentication problem, the authenticated marking scheme [22, 34], an extension of advanced marking technique, adds functionality for the end host to authenticate incoming marked packets. If a marked packet fails to be authenticated, it will be dropped. Therefore, the false marked packets which are generated by malicious users cannot prevent end hosts from discovering routing paths.

Even though the advanced/authenticated marking scheme solves many issues of the original edge sampling technique, it still introduces extremely high overhead into marked packets. In order to discover routing paths efficiently, this scheme requires the sampling probability on the order of 1/25 to 1/10. This means that on average for every 25 forwarded packets one packet (marked packet) has to suffer processing delay before it can be forwarded. Overall, the overhead is still too high. Another big concern is that the probabilistic packet marking scheme (including node sampling, edge sampling, and advanced/authenticated sampling) is not able to handle reflective DDoS [35] attacks. Since this scheme encodes path information into the marked packets and these packets are only forwarded to a destination host, the victim can only receive marked packets from routers between reflectors and victim: there is no way for the victim to receive any marked packet from the true attack sources, which are beyond reflectors. Therefore, it is impossible for the victim to trace the true attacker(s) in reflective DDoS attacks. Figure 2.1 explains the problem in tracing reflective DDoS attacks.

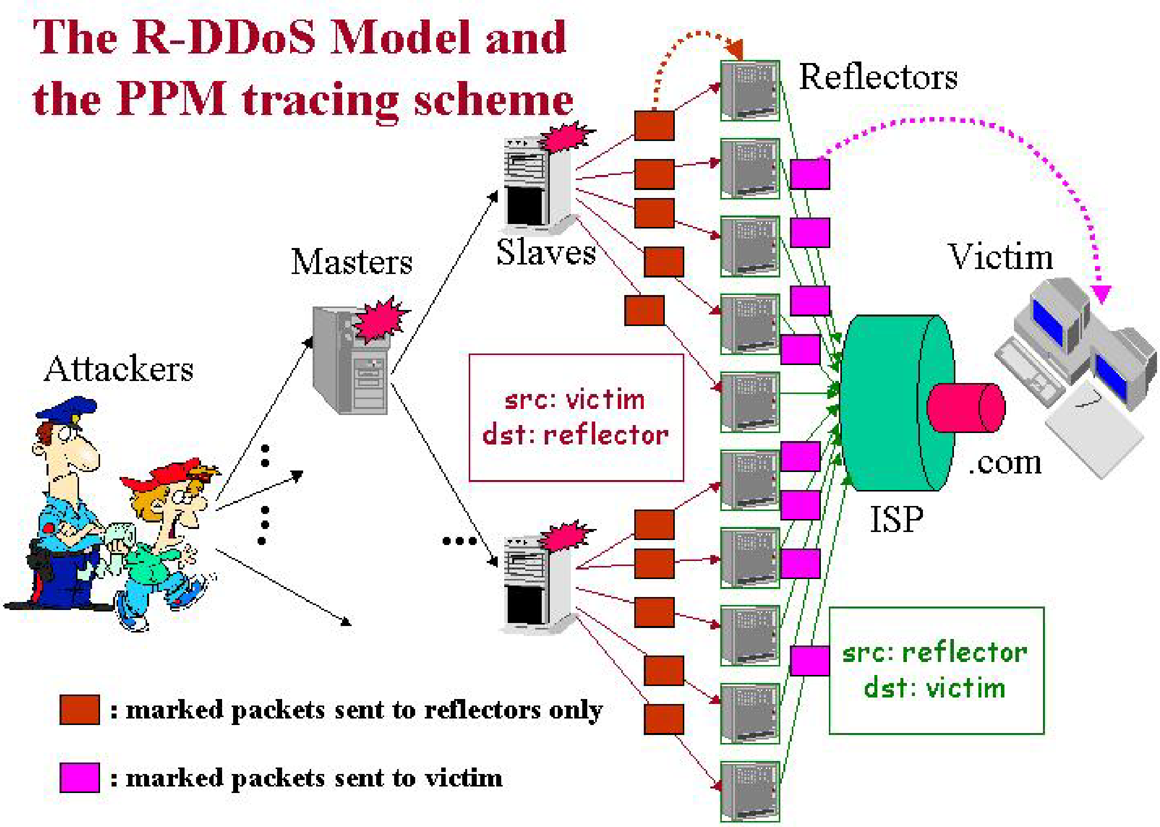


Figure 2. 1 – The Reflective DDoS attack model and PPM tracing scheme: the PPM scheme is not able to trace reflective DDoS attacks.

2.4.2 ICMP traceback (iTrace) scheme

The ICMP Traceback (iTrace) [17] is another type of sampling-based tracing schemes, proposed by S. M. Bellovin in 2000. The iTrace scheme randomly selects and copies a packet in probability ***p*** and then the original packet is forwarded immediately; as a result, the selected packet will not suffer too much processing delay. Once a packet is selected, a special ICMP packet, called iTrace message or iTrace packet, is generated and sent to the host which is the destination address in the IP header of the selected packet. This message contains useful information such as the pair of IP address and MAC address of both the upstream and downstream links of the router that selected the packet. In addition, the selected packet's source and destination addresses and a portion of payload are included for the end host to analyze in its endeavor to discover attack paths. Last, the iTrace message contains optional authentication messages to prevent a malicious user from forging iTrace messages.

Until today the iTrace approach is the only proposed scheme that can handle reflective DDoS attacks [35]. Since this scheme copies the source and destination IP addresses from the selected packet, it is possible to send the iTrace messages for both the source and destination hosts, instead of sending iTrace messages for only the destination machine. Therefore, the victim is able to receive iTrace messages from routers beyond reflective routers and locate the true attacker(s). Figure 2.2 illustrates how the iTrace scheme solves the tracing problem of reflective DDoS attacks. The detailed discussions about iTrace are presented in Chapter 4.

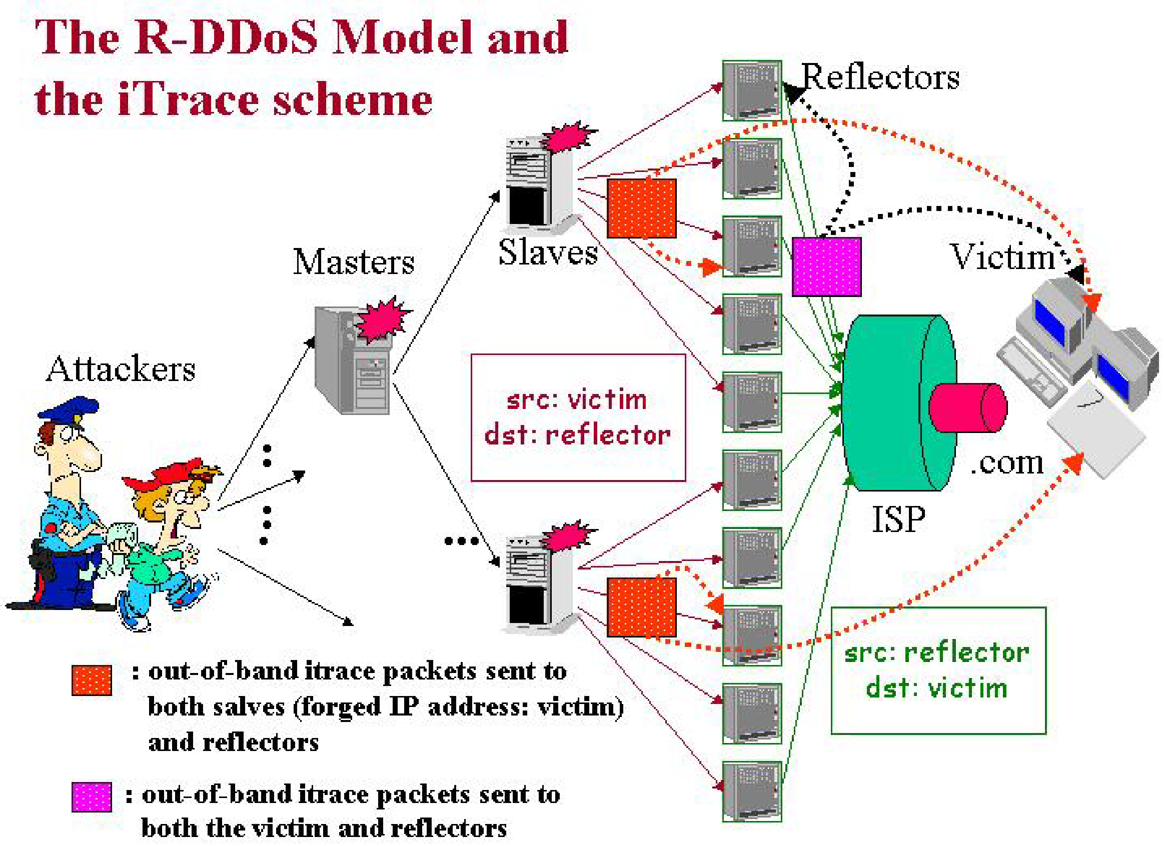

The R-DDoS Model and
the iTrace scheme
Reflectors
Masters
Slaves
Victim
Attackers
src: victim
dst: reflector
.com
ISP
src: reflector
dst: victim
🟥 : out-of-band itrace packets sent to both salves (forged IP address: victim) and reflectors
🟪 : out-of-band itrace packets sent to both the victim and reflectors

Figure 2. 2 – The Reflective DDoS attack model and the iTrace scheme: the iTrace scheme can trace reflective DDoS attacks.

2.5 Summary of this chapter

In this chapter, we reviewed and analyzed several proposed solutions of IP traceback issues. From these reviews, we realize that when we design a tracing mechanism, how to maintain network performance is as important as how to trace attacker(s). For example, the ingress filter technique can reduce problems of forged IP; however, many ISPs will not adopt it because of the concern of degrading network performance. How to trace attacker(s) efficiently without sacrificing network performance is today's research focus in solving IP traceback problems. Current research proposals showed us that sampling packets in probability ***p*** to provide tracing information, instead of checking every single packet, might be a good approach to deal with both tracing efficiency and network performance.

Since DDoS and reflective DDoS attacks have become the method of choice for attackers, how to handle these types of attacks is also a major concern in devising traceback schemes. Within today's proposed solutions, only the ICMP traceback (iTrace) scheme, one of the sampling approaches, can solve the tracing problem of DDoS and reflective DDoS attacks. Therefore, we choose the iTrace scheme as a base to solve the IP traceback problem.

However, the iTrace scheme is still not able to provide a complete solution. In the distributed style of attacks, to sample attack packets near the victim is easy because there are huge volumes of attack packets; but to sample packets near the attacker may be difficult because of the small amount of attack packets. This limitation of the iTrace scheme can be solved by incorporating the IPSec-based tracing scheme into the solution, since the IPSec protocol is widely employed in today's internet. The traceback scenario could be: utilize the iTrace scheme to narrow down the suspected attack domains and use the IPSec-based tracing scheme to locate the true attack source(s) in each suspected attack domain. In this dissertation, we will focus on both IPSec-based and iTrace-based approaches to understand both schemes' advantages and limitations, and propose our enhancements to solve the IP traceback problem effectively.

Chapter 3 IPSec-Based Source Tracing

Node-to-node authentication provides a means for us to solve the principal problem in tracing attack sources. Through the authentication mechanism, if one node receives an authenticated packet, this node can be sure that the authenticated packet is forwarded by another node. Thus if we set up an authentication tunnel (virtual tunnel) between a victim and a router and the victim still receives attack packets which are authenticated, we can be certain that this router belongs to an attack path. By repeating this process, ultimately we are able to trace attack sources and then stop attacks.

The IPSec protocol provides node authentication functionality as a default. Since the IPSec protocol suite [16] has been standardized and is widely employed, it is convenient for us to develop a tracing mechanism on top of the IPSec infrastructure. The Decentralized Source Identification for Network-based Intrusion (DECIDUOUS) [21] is a tracing mechanism which is developed by utilizing node authentication of the IPSec. In this chapter, we present the design of the DECIDUOUS tracing mechanism and investigate the limitations of this approach in tracing attacks across different administration domains. Then the Packet Head Information List (PHIL) and the PHIL-switching techniques are proposed and developed to enhance the tracing ability of DECIDUOUS.

3.1 Brief introduction to IPSec protocol

The IPSec protocol [16, 36] provides security functionality, including authentication and encryption at IP level. Any IP packets which are required to be authenticated or secured can utilize the IPSec to provide security services. There are three principal components in IPSec suite:

- The Authentication Header (AH) protocol [26] provides only the function of authentication to IP packets.
- The Encapsulating Security Payload (ESP) protocol [27] provides the encryption and the authentication (option) to IP packets.
- The Key management [37] provides the negotiation of connection parameters, such as the key, the communication protocols (ESP or AH), and the algorithms (MD5, HMAC-MD5, 3DES, etc.)

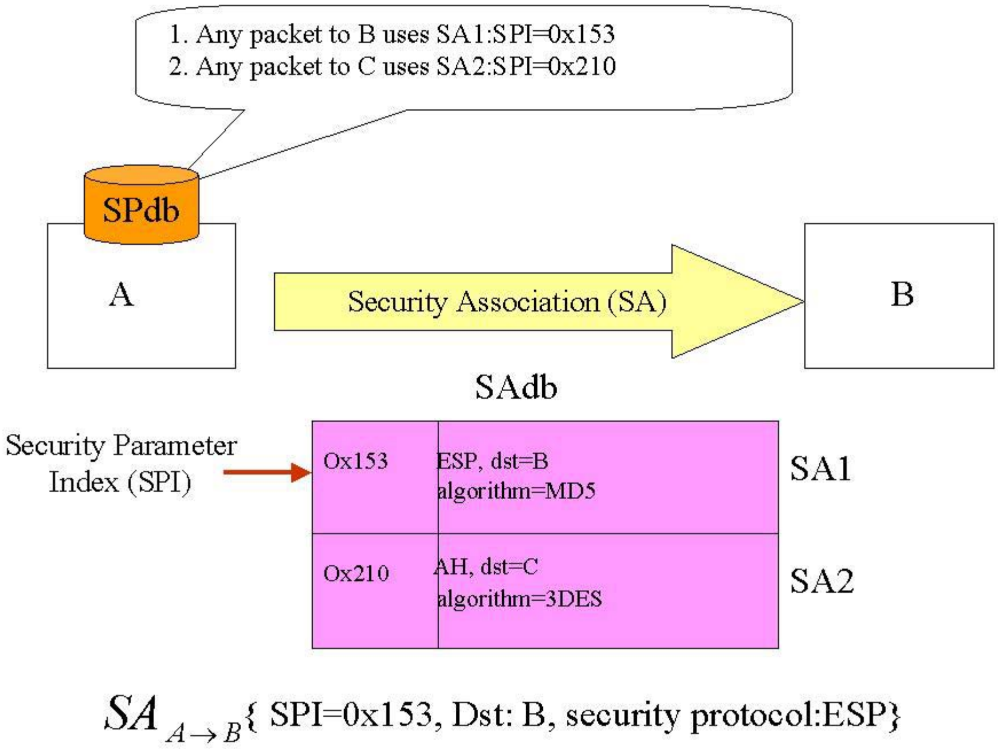

1. Any packet to B uses SA1:SPI=0x153
2. Any packet to C uses SA2:SPI=0x210
SPdb
A
Security Association (SA)
B
SAdb
Security Parameter Index (SPI)
Ox153
ESP, dst=B
algorithm=MD5
SA1
Ox210
AH, dst=C
algorithm=3DES
SA2
$SA_{A \rightarrow B}$ { SPI=0x153, Dst: B, security protocol:ESP}

Figure 3. 1 – The Security Association (SA)

The important concept of the IPSec [16] is the security association (SA) between two network entities. An SA is a one-way relationship that affords security service to traffic. As shown in Figure 3.1, if node A wants to build up an IPSec tunnel with node B (A → B), A and B should negotiate an SA to describe security protocols (AH or ESP) and algorithms to authenticate or encrypt data. All SAs are put into the SA database (SAdb).

How to process IP packets depends on security policies, which are stored in the security policy database (SPdb). For example, if an incoming packet matches one of the policies in node A (see Figure 3.1), a related SA (security protocols and algorithms) from the SAdb will be used to authenticate or encrypt this packet. If this packet does not match any policy, it will be forwarded as normal traffic.

Node authentication, one function of the IPSec suite, provides a mechanism to trace attack sources. If there is an IPSec tunnel from node A to node B, a trusted relationship between node A and node B can be maintained. For example, if B received an IP packet from this IPSec tunnel (A→B), one thing can be sure that this IP packet was really forwarded by A. If B can set up IPSec tunnels with other upstream nodes in the same way, B is able to identify where the attack packets are coming from, and thereby locate and stop attacks. The DECIDUOUS [23, 40] tracing module utilizes this technique to discover attack paths and locate attack sources.

3.2 The principal problem of the IPSec protocol

Although the IPSec can provide node authentication, the IPSec header information (secure information) will be dropped after the IPSec process is completed (See Figure 3.2.); as a result, most applications above the IPSec-layer have no way of obtaining secure information. However, such information is extremely useful to applications like Intrusion Detection Systems (IDS) [39, 40] for identifying attacks and to applications like SNMP [41, 42] for network management. Given the state of current IPSec implementations, there is a need to bridge the big gap between applications and IPSec protocol

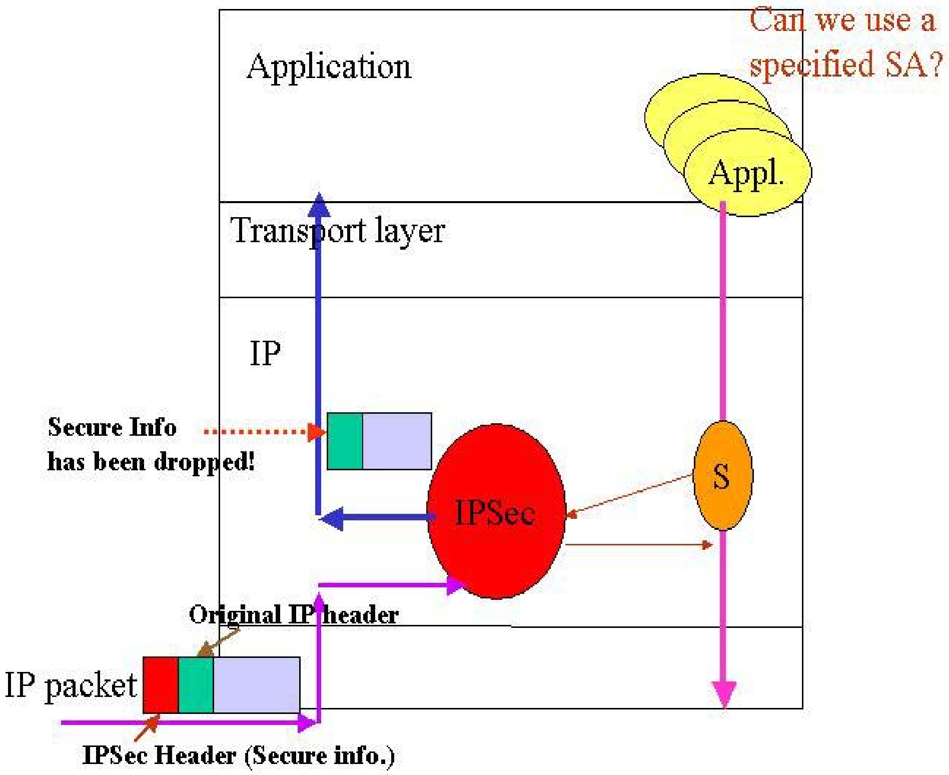

Application
Can we use a specified SA?
Appl.
Transport layer
IP
Secure Info has been dropped!
IPSec
S
Original IP header
IP packet
IPSec Header (Secure info.)

Figure 3. 2 – The IPSec header is dropped after IPSec process.

3.3 The DECIDUOUS traceback approach

DECIDUOUS [21, 38] is a tracing mechanism which utilizes IPSec tunnels to identify attack sources. The premise of this approach is that, if a packet is correctly authenticated by a certain router, it must have been forwarded or generated by that router. Using this idea, DECIDUOUS can establish a series of IPSec tunnels with upstream routers and cooperate with IDS to identify attack sources. The DECIDUOUS operation is based on two principal assumptions. First, the Intrusion Detection System (IDS) will trigger the DECIDUOUS daemon to dynamically set up authentication tunnels. It is noted that the DECIDUOUS itself cannot detect attack packets. However, by setting up IPSec tunnels and collaborating with IDS, the DECIDUOUS tracing scheme is able to discover where the attacks are coming from. Another assumption is that DECIDUOUS should have enough knowledge of network topology to decide where to set up IPSec tunnels. Within a local administration domain, it is easy for DECIDUOUS to know the entire local network topology. However, in order to trace attacks across different administration domains, the DECIDUOUS daemon has to know at least a border gateway of other domains to establish SAs.

3.3.1 The DECIDUOUS design

The DECIDUOUS module is designed to establish IPSec tunnels flexibly and dynamically and then locate attackers with the help of IDS. To achieve this goal, DECIDUOUS introduces the concept of CutSet. A CutSet is a group of routers which are separated from the victim by the same hop count. With the concept of CutSet, DECIDUOUS is able to transform the network topology into a linear topology [40]; thus, DECIDUOUS is able to establish IPSec tunnels among different CutSets and work with IDS to identify attack sources. Figures 3.3 (a) and 3.3(b) illustrate the CutSet and the detection algorithm for DECIDUOUS to trace attacks.

As an intrusion detection system (IDS) is aware of attacks, it will signal the DECIDUOUS daemon; and as a result, DECIDUOUS will set up IPSec tunnels from routers (within the same CutSet) to victim with a unique security parameter index (SPI) for each tunnel. After DECIDUOUS has set up these tunnels, IDS will verify the attack again, based on the authentication information provided by these tunnels, and then send the result to DECIDUOUS which will decide whether the attack sources are within or beyond this CutSet. If the attacker is within this CutSet, DECIDUOUS is able to discover the attack source by examining the unique SPI. If the attack is beyond this CutSet, DECIDUOUS will destroy the IPSec tunnels already built and set up other tunnels with other CutSets.

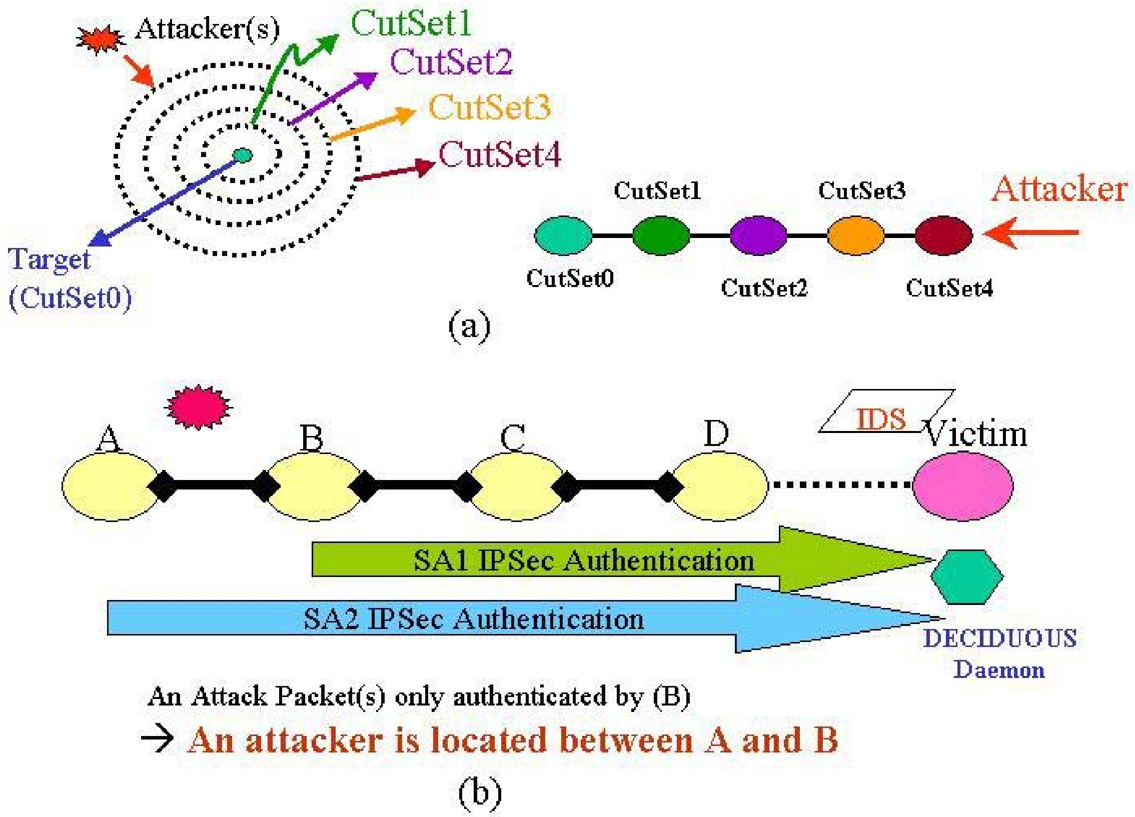


Figure 3. 3 – (a) The concept of CutSet (b) The detection algorithm.

The following are the algorithms (pseudo codes) for DECIDUOUS to discover attack sources and for IDS to detect attacks.

The pseudo codes for DECIDUOUS to set up SA and locate attack sources

```
Int Deciduous_Setup_SA
{
  /* Input:  1. IDS_ trigger&Signal.  2. Network topology
             3.Transform_NetworkTopology_to_LinearCutSet (NetworkTopology);  */

                If (IDStrigger & Signal)
                {
                        while(CutSet!=NULL)
                        {       /* start from the nearest CutSet */
                                Establish_SA_FromCutSet2target( );
                                /* inform IDS with the SA information*/
                                SynchronizingWithIDS( );
                                Waitingfor_SignalFromIDS( );
                                If (Result ==Attack) Locate_AttackSource( );
                         }
                 }
                return (DECIDUOUS_FINISH);
}
```

The pseudo codes for IDS to detect attacks and report the results for DECIDUOUS

```
Int IDS_Detection_attacks
{
        switch (DetectionStatus)
        {
                case Without SASetup:
                      Signal_DECIDUOUS2Setup_SA( ); break;
                case  SAHasBeenSetup:
                      if (isAttackPacket){Signal_DECIDUOUS("AttackPacket", SPI=xxx");}
                      else {Signal_DECIDUOUS("NoAttackPacket");}
                      break;
         }
         return (IDSDetection_FINISH);
 }
```

3.3.2 Challenges for DECIDUOUS module

Given the state of current network environment, the DECIDUOUS tracing scheme faces two challenges. First, since DECIDUOUS is developed on top of the IPSec infrastructure and utilizes the IPSec header information to locate attack sources, how to obtain the IPSec security header for tracing attacks is a big issue. On the one hand, if the attack can be detected in the network layer, the IDS can be integrated into the network protocol stack and can obtain the security information directly, but on the other hand, if the attack can only be detected above the network layer, the IDS is definitely unable to obtain the original IPSec authentication header because the IPSec header information has been dropped after the IPSec process is completed (for the incoming side) [28, 29]. Second, regarding security concerns, different administration domains are not able to share local network topologies with others. How to set up IPSec tunnels among inter-domain environments is another challenge for the DECIDUOUS tracing model.

To overcome these challenges, the Packet Header Information List (PHIL) and PHIL-API suite are proposed to allow the application-level IDS to access IPSec header information [28]. Also the PHIL-switching technique is developed to extend the capability of DECIDUOUS to support inter-domain tracing. The PHIL/PHIL-APIs and the PHIL-switching designs are presented in the following sections.

3.4 PHIL and PHIL-APIs

The techniques of PHIL and PHIL-APIs aim at solving the first challenge for the DECIDUOUS tracing module. PHIL provides a data structure to store IPSec headers when the IPSec process discards them. PHIL-APIs provide convenient socket interfaces such that applications are able to receive both data and PHIL (IPSec headers). For sending data, PHIL-APIs provide options for users to send both data packets and preferred SAs (authentication algorithms and protocols) for IPSec to process these outgoing packets.

This technique not only solves the first problem of DECIDUOUS but also bridges the big gap between IPSec and applications. As a result, any application which requires security services can use IPSec to authenticate or encrypt data packets. Thus, the PHIL/PHIL-APIs can reduce the duplication of security services at different layers, thereby simplifying the design of applications.

3.4.1 Motivations

In current IPSec implementations, the important authentication header is discarded when the IPSec process is finished; as a result, applications above IP layer are not able to obtain the security information from the received data. Moreover, IPSec does not allow applications to access a set of security attributes for sending packets. These issues, which are major concerns for the DECIDUOUS tracing scheme, motivate us to study the glaring deficiency of the IPSec protocol and propose a practical solution--PHIL/PHIL-APIs.

3.4.2 The design and Implementation of PHIL and PHIL-APIs

To bridge the gap between applications and IPSec, the Packet Header Information List (PHIL) and PHIL-APIs are proposed to provide a socket mechanism (PHIL_APIs) for applications to receive not only data but also IPSec headers (PHIL) from incoming packets. This can be achieved by retaining the security headers while processing the incoming traffic and passing them to the transport and application layers. Applications which need the PHIL information can use PHIL-APIs (phil_enable( ) and phil_recvfrom( )) to retrieve the PHIL information. For the outgoing side, more parameters as well as a socket mechanism (phil_sendto( )) are added. In Figure 3.4, before the IPSec process discards the header information, PHIL retains this important information from the incoming packets. On the application level, the PHIL-API suite supports the user-level system calls to access the incoming packets and the IPSec header information. For the outgoing packets, one can send a query to the security association database (SAdb) to obtain available SAs. With the query results and the PHIL-APIs, applications not only can send the data but can also request a preferred SA for authenticating and/or encrypting outgoing data.

The capabilities of the PHIL-APIs are as follows:

- For incoming traffic, the PHIL-APIs provide socket interfaces such that the application developers are able to retrieve IPSec headers.
- For outgoing traffic, the PHIL-APIs provide a functionality to interact with the kernel's Security Association database (SAdb) and the Security policy database (SPdb) to obtain available SAs. Thus, applications can use PHIL-APIs ((phil_sendto( )) to send out packets using these SAs.

The current implementation of PHIL/PHIL-APIs is in Linux kernel version 2.0.36 [43] with FreeSwan.1.0 package [44]. The details of PHIL and the PHIL-API extension can be found in [28, 29].

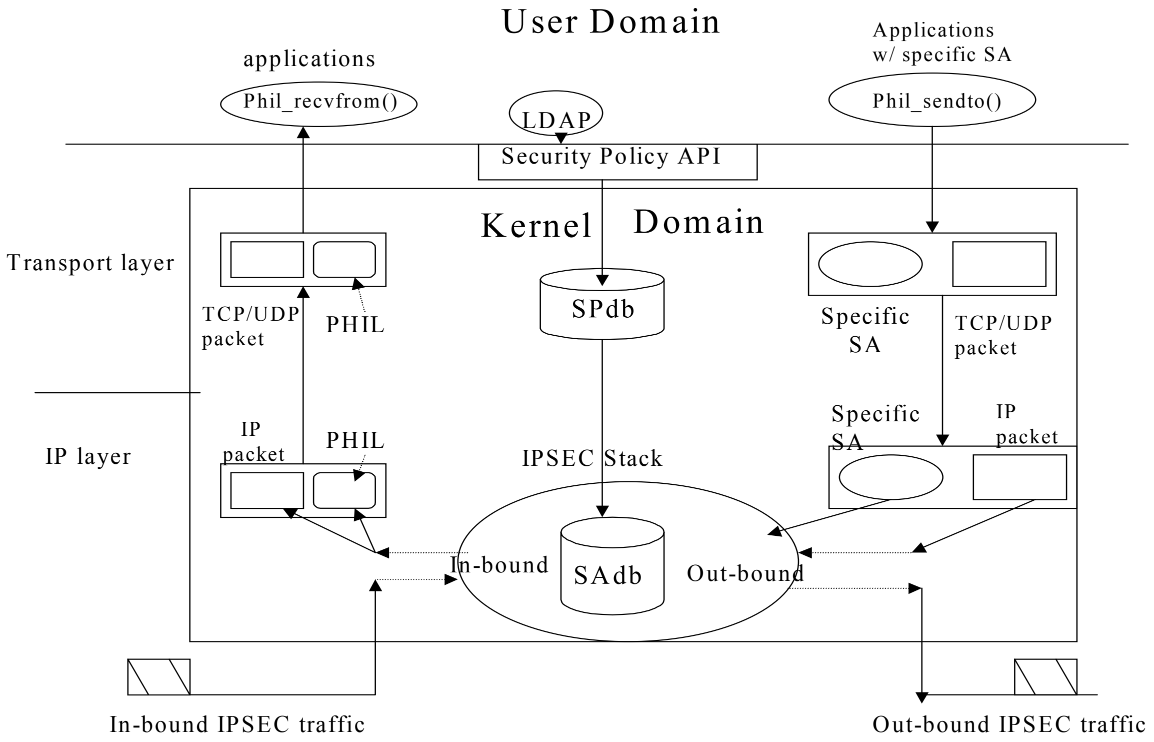


Figure 3. 4 – PHIL and PHIL-API design concept

3.4.3 PHIL-API Suite

The PHIL-API suite can be categorized as follows: PHIL-APIs for receiving traffic, PHIL-APIs for sending traffic, and the query functions.

I. There are three PHIL interfaces for users to receive data and PHIL:

```
a. int phil_enable( int sockfd, int mode )
   int phil_disable( int sockfd )
```

A TCP [45] or UDP [46] socket opened with a socket system call should first be enabled to receive the PHIL information along with the application data. Users can disable the PHIL function as well.

Input: It takes the socket descriptor value (sockfd) and integer mode as input parameters

Returns: It returns a non-zero error value if the call failed, otherwise it returns a value ZERO.

Description: phil_enable( ) prepares a socket to receive the PHIL information. The mode parameter can either be EXHAUSTIVE or NON_EXHAUSTIVE. If the receive mode is set to NON_EXHAUSTIVE, the PHIL information consists of only the SPI value associated with each data segment. In the EXHAUSTIVE mode, the PHIL information consists of the entire set of security parameters associated with the packet. This function call must be used before any other extended API calls. After enabling a socket with PHIL, an application can still use normal socket API calls to receive/send data. However, PHIL-APIs must be used if someone wants to receive/send data and the PHIL.

b. int phil_accept(parameters to accept(), char *phil_buf,int phil_len)

Input: The parameters of the corresponding standard socket API call accept(), and the buffer name (phil_buf) and its length (phil_len).

Returns: In addition to the return values of the corresponding normal API call accept(), this call returns phil_buf, which is a character buffer that contains the PHIL information.

Description: A phil_accept() call is used by the operation of TCP servers in the concurrent mode. The phil_accept() call used over the parent socket will provide information about the peer party, which can be trusted based on the PHIL information. A server can then decide whether it should accept the new request, based on some policies.

c. int phil_recvfrom(parameters to recvfrom(), char *phil_buf, int phil_len, int *dsegs)

Input: The parameters are the corresponding standard socket call--recvfrom(), the buffer name (phil_buf), its length (phil_len), and integer pointer dsegs.

Returns: In addition to the return values of the corresponding normal system call recvfrom(), this call also returns the phil_buf buffer, and the integer value dsegs is the number of TCP data segments that constitute the total data bytes that were read from this call.

Description: A phil recvfrom() call is used in place of a recvfrom() call to retrieve data plus the PHIL. This call is used by both UDP and TCP applications.

II. Two PHIL-API function calls for sending data are presented.

a. int phil_bind(int sockfd, unsigned long *spi array, int size)
int phi_unbind(int sockfd)

For the outgoing data, the phil_bind() can directly bind one or more SAs to a socket and phil_unbind() can undo this binding.

Input: Socket descriptor value, the SPI array, and its size.

Returns: ZERO on success and non-zero on failure.

Description: The system call phil_bind() merely records a possible set of SPIs that could be used by this socket. The specific values of SPIs for each sending process can be different and should be specified at the time of phil_bind().

b. int phil_sendto(parameters to sendto(), long *spi_arr , int size)

Input: In addition to the corresponding normal API call sendto(), an SPI array and its size are specified.

Returns: Number of bytes sent on success or a negative error value.

Description: The spi_arr array describes the preferred SPI values to be sent. If an application is not aware of the possible SAs/SPIs, it can send a query to the SAdB to obtain the SPI values through the query_spi() call. The value of the SPIs should be a subset or equal to the SPI values that were bound to the socket using the phil_bind() call. If only one spi among all the spi's was bound, the IPSec process will be applied to outgoing packets. The spi field can be used to specify a SA or to choose a SA among several possible SAs. Through this call, it is possible to multiplex the data from an application over several SAs and provide different levels and features of security to different data types. Here it is emphasized that the applications' preference of sending its data over a SA will be honored only if the applications conform to the SPDB. In case of conflict between SPDB and an applications' preference, the data will not be sent and the user should be notified.

III. Two query functions are implemented as follows:

a. int phil_ qspi (struct sockaddr* src, struct sockaddr * dst, char * buf, int size, int *num)

From phil_bind( ), an application might want to know the availability of SAs and their SPI values before it sends data. The phil_qspi( ) satisfies such a request.

Input: Source and destination addresses and size of the buffer to hold the spi values.

Returns: ZERO on success and non-zero error value on failure. On success, this function call returns the spi values in the buffer and the number of available spis in the integer num.

Description: Because the SAs are simplex in nature, a query is sent to the SAdb to check the availability of a specific SA. It is noticed that in the case of VPNs the source IP address is most likely to be the security gateways of VPNs. The security protocol and the spi value returned in the spi_struct will help the application to select a SA.

b. int phil_qsadb(unsigned long spi, struct sockaddr *dst, char * buf, int size)

This function is used to send a query to the SAdb for detailed information about the SA that is indexed by the spi value, which is passed as a parameter.

Input: a selected spi, destination addresses, the pointer of the buffer, and size of the buffer to hold the spi values.

Returns: ZERO on success and non-zero error value on failure. On success, the SA information and the data size of the SA are also returned.

Description: When an application sends data, it first sends a query to the SAdb to find out whether a preferred SA exists. If the SA is in the SAdb, detailed SA information --- protocols and algorithms -- will be put into the buffer and returned to this application.

With the support of the PHIL_API suite, developers can flexibly use phil_sendto() with a preferred authentication/encryption protocol and security level to authenticate/encrypt data , and use phil_recvfrom() to receive data and SA information.

3.4.4 The Evaluation of the PHIL implementation

In this section, we present the evaluation of the overhead (delay) introduced by PHIL. The major delay caused by PHIL is the PHIL extracting process from the incoming side and the PHIL matching-and-replacing process on the outgoing side (see Figure 3.5).

The configuration for PHIL evaluation, as shown in Figure 3.5, includes two machines: both are 450 MHz Pentium II personal computers equipped with 10 MB/sec Ethernet cards. The echo client/server (TCP and UDP) program is developed to measure the average delay. (It is noted that the delay is dominated by the IPSec process and the request/reply processing time of CPU. The maximum delay could be actually caused by the IPSec process and the request/reply process and not by the PHIL process. Thus measuring the maximum value is meaningless for the PHIL evaluation. In this evaluation we just focus on the measurement of the average delay and ignore the maximum delay.)

In each test, 100,000 packets are launched and the average time is calculated and recorded. Various data sizes in 16, 32, 64, 128, 256, 512, 1,024 bytes are also tested to evaluate the effect of data size. Since any packet greater than the maximum transmission unit (MTU) will be fragmented, the PHIL process could be executed for more than two times, depending upon the data size in each packet. Hence, the packet size is controlled (not to exceed 1500 bytes) to avoid calculating the overhead twice.

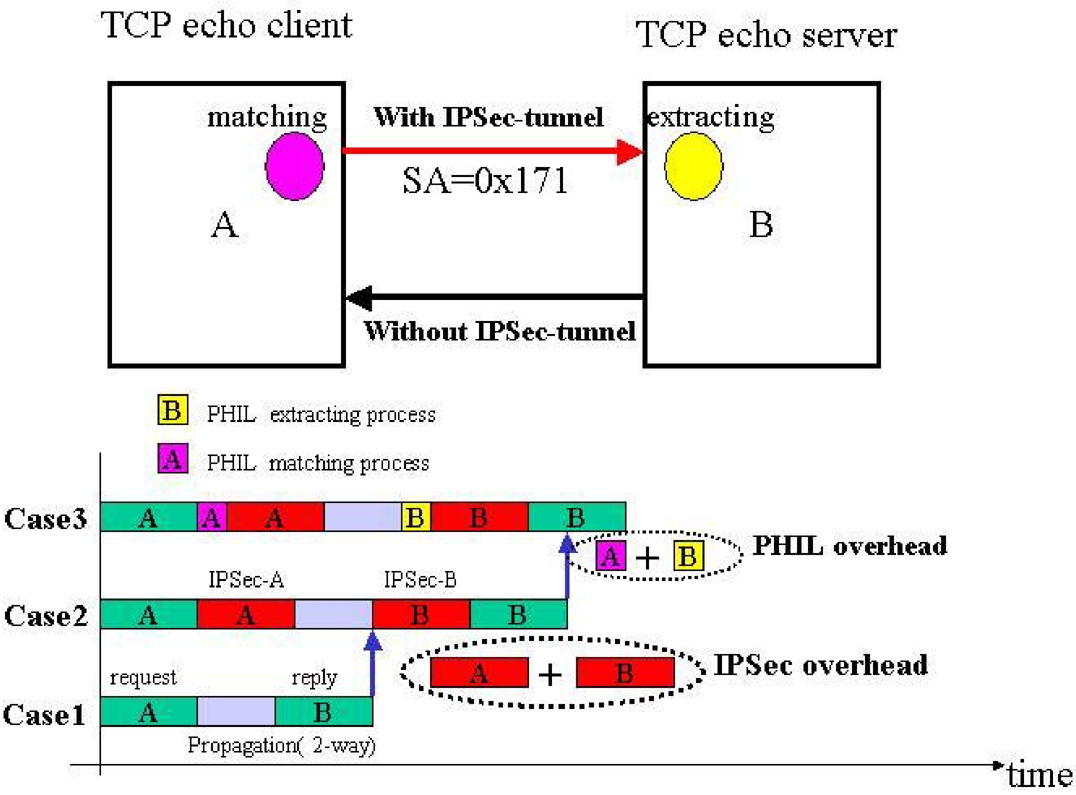


Figure 3. 5 – The configuration for PHIL evaluation and PHIL testing plan.

There are three test cases to evaluate the average delay:

Case 1: Using the Linux 2.0.36 kernel only.

Case 2: Using the Linux 2.0.36 kernel with the FreeS/wan IPSec package, no PHIL process.

Case 3: Using the Linux 2.0.36 kernel with the FreeS/wan IPSec package and the PHIL process.

In each case, both protocols (ESP and AH) are tested. The results are shown in Table 3.1. From case 1 and case 2, the overhead caused by IPSec is calculated. From case2 and case3, the overhead introduced by PHIL is computed. (Please see Figure 3.5.)

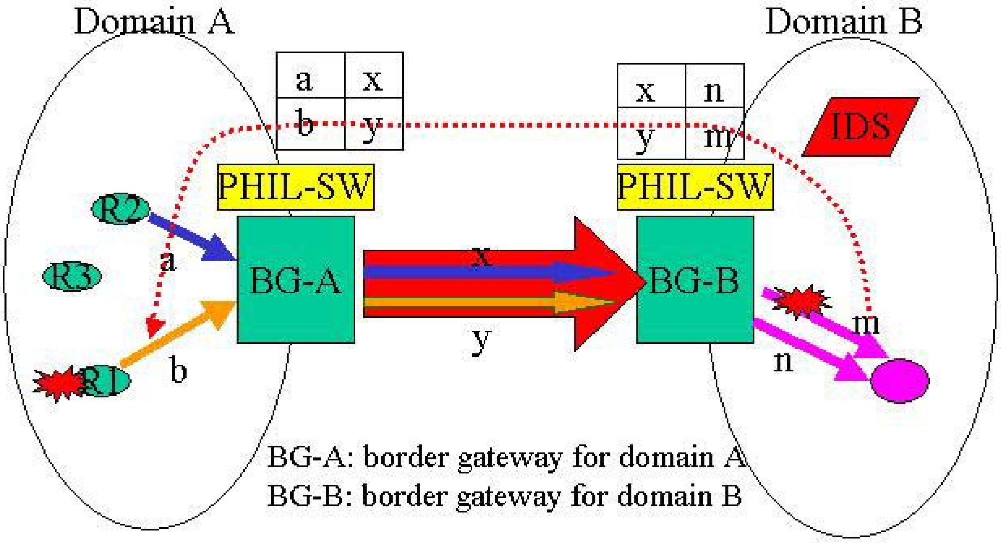
Table 3. 1: The results of evaluating IPSec/PHIL overhead

size(bytes)	Case 1	Case 2	Case 3	PHIL overhead (case3 – case2)
16	363 us	ESP: 505 us	ESP: 506 us	ESP: 1 us
		AH: 502 us	AH: 502 us	AH: 0 us
32	398 us	ESP: 538 us	ESP: 539 us	ESP: 1 us
		AH: 535 us	AH: 536 us	AH: 1 us
64	465 us	ESP: 616 us	ESP: 617 us	ESP: 1 us
		AH: 613 us	AH: 613 us	AH: 0 us
128	601 us	ESP: 761 us	ESP: 762 us	ESP: 1 us
		AH: 756 us	AH: 757 us	AH: 1 us
256	870 us	ESP: 1050 us	ESP: 1051 us	ESP: 1 us
		AH: 1045 us	AH: 1046 us	AH: 1 us
512	1410 us	ESP: 1628 us	ESP: 1630 us	ESP: 2 us
		AH: 1625 us	AH: 1625 us	AH: 0 us
1024	1510 us	ESP: 1812us	ESP: 1814us	ESP: 2 us
		AH: 1809 us	AH: 1809 us	AH: 0 us

We concluded from analyzing the results that no matter what the data size (less than MTU) is, the average overhead introduced by PHIL is almost constant (1-2 microsecond) because the extracting process and matching process are independent of the data size. (Please note that the data size increase only causes the overhead of IPSec process to increase.) In our test cases, we eliminated the effect of IPSec delay, which is not the overhead of PHIL process, by subtracting the results of test case 3 from the results of case 2.

3.5 PHIL switching

The PHIL-switching technique is proposed to solve another problem posed by DECIDUOUS: how to trace attacks across different domains. Instead of requesting other domains to share their network topologies, one can utilize the PHIL-switching scheme to correlate the security information among different administration domains. The premise of the PHIL-switching scheme is that the relationships between SAs among peer domains are maintained in the PHIL-switching table. If someone in an administration domain wants to know the relationship of IPSec tunnels with another domain, he can send a query to the PHIL-switching tables of these two domains to obtain the requested information. With the PHIL-switching support, one can trace attack sources across different domains by sending queries and analyzing the relationships among IPSec tunnels in different domains. For example, Figure 3.6 illustrates how DECIDUOUS works with the PHIL-switching scheme to trace attacks across two administration domains. If the intrusion detection system (IDS) within domain B detects an attack from the tunnel m, this result will be passed on to the DECIDUOUS daemon of domain B. The DECIDUOUS daemon in domain B looks up its PHIL-switching table and correlates the attack flow with the incoming tunnel y. Then the DECIDUOUS daemon in domain B will inform the DECIDUOUS daemon of domain A about the attack flow (tunnel y). The DECIDUOUS daemon of domain A, after receiving this information, can simply look up its PHIL-switching table and correlate the attack with the tunnel b. Ultimately DECIDUOUS is able to locate the attack source at R1. It is noted that without the need to explore any network topology, the DECIDUOUS tracing scheme is able to trace attack sources between these two domains by working with the PHIL-switching technique.



BG-A: border gateway for domain A
BG-B: border gateway for domain B

1. IDS: detect attack packet from tunnel m.
2. DECIDUOUS at B: table lookup and associate the attack with the tunnel y.
3. DECIDUOUS at B: pass this information to PHIL-SW at A
4. DECIDUOUS at A: table lookup and associate the attack with the tunnel b.
5. Domain A locates the attack at R1.

Figure 3. 6 – DECIDUOUS tracing module with PHIL-switching support

3.5.1 The capabilities of the PHIL-switching

The capabilities of the PHIL-switching design are as follows:

- For any incoming IPSec traffic, if it matches any entry in the PHIL-switching table, it will be forwarded to a specific IPSec tunnel described in that entry.
- If the incoming traffic is a non-IPSec packet, it may be processed as normal IP traffic, i.e. the incoming traffic still can be forwarded but without being put into the IPSec tunnel.

- ❑ Traffic from different tunnels can be aggregated into one tunnel (IPSec traffic aggregation).
- ❑ The PHIL-switching provides filter functions to drop or switch packets regarding the PHIL-switching policies.
- ❑ The PHIL-switching provides interfaces for users to control the PHIL-switching table.

3.5.2 The design and implementation of the PHIL-switching

The PHIL-switching module includes two components:

- PHIL-switching controller: The PHIL-switching controller provides interfaces for users to add, delete or flush the PHIL-switching table. In the current PHIL-switching implementation, the controller can provide two types of interfaces: the SNMP model and the client/server model using UDP. In the SNMP model, one can control (read or write) the PHIL-switching table through the SNMP mechanism. In the client/server model, users can remotely control the PHIL-switching table through client-server communication. Users can control the switching table in the following ways:

  "Read" the PHIL switching table,

  "Add" one entry to the PHIL switching table,

  "Delete" one entry from the PHIL switching table,

  "Flush" the whole PHIL switching table.

- Switching entity: The switching entity provides switching functionality in the PHIL-switching mechanism. For any incoming packet with fields [SPI, security protocol, source/destination address, protocol, source/destination port], the PHIL-switching daemon will look up the PHIL switching table, and then switch this packet into different tunnel(s) using specified SPIs, which is defined in the PHIL switching table. If this packet does not match any entry, it will be forwarded as normal traffic.

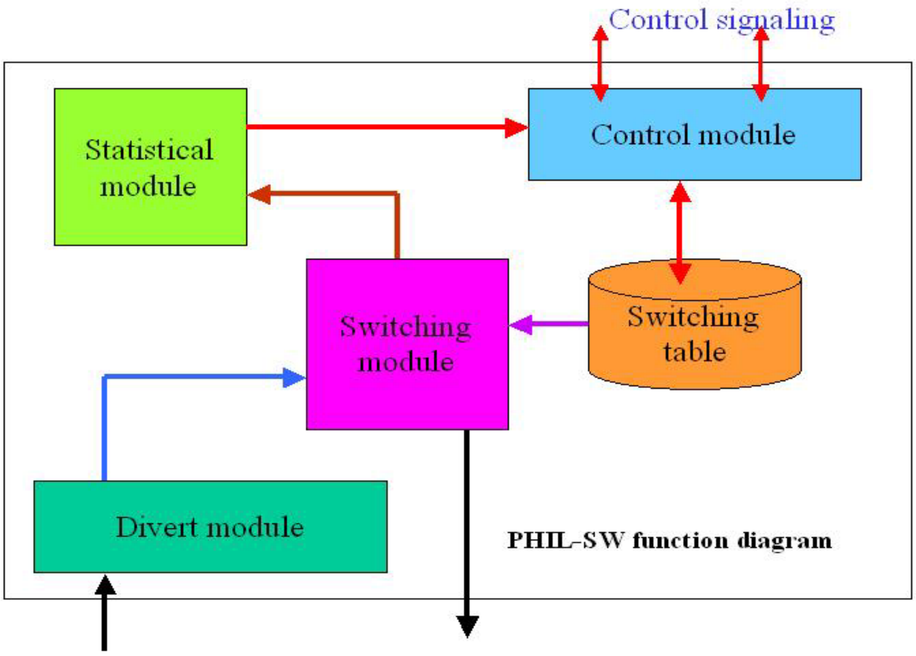


Figure 3. 7 – PHIL-switching function block.

The function blocks of the PHIL-switching are presented in Figure 3.7. The control module provides interfaces-- the SNMP model or the client/server model--for users to control or access the PHIL-switching table. The statistical module provides statistical information to monitor the traffic through the PHIL-switching. The divert module provides a mechanism to intercept desired packets from kernel domain. The switching module switches or drops the packets, depending on the switching policies.

In the current stage, we implemented the PHIL-switching functionality on the user domain because it is flexible: we can tune the PHIL-switching daemon without having to recompile and reboot the kernel. Thus a Linux-based divert socket is applied to intercept IP packets from kernel domain to user domain. The Linux divert socket [47] has been designed and implemented as part of a DARPA-funded network security project on ANR (Advanced Network Research) of MCNC. It is open source and free downloadable.

3.5.3 The evaluation of PHIL-switching implementation

The configuration, shown in Figure 3.8, is set up to evaluate the overhead (delay) caused by the PHIL-switching. There are three test cases in this evaluation. In case1, we set up one-way IPSec tunnels between A and B and B and C. In case2, we used the divert socket to obtain the incoming IP packet and then to re-inject the diverted IP packet into the kernel. In case3, we used the divert socket to get the incoming IP packet and switched this packet by using the phil_sendto( ) with specified SPI/SA to push the data back into the kernel. In each case, we sent 10,000 packets to measure the overhead and recorded the average delay (total time/10,000). (Since the maximum delay might be caused by IPSec process or divert socket process, one snapshot of maximum measured value is meaningless to evaluate the maximum delay of PHIL-switching. Thus we focus on only measuring the average delay and ignore the measurement of maximum delay.) The overhead in the test cases includes the divert-socket delay caused by divert intercepting process and PHIL switching process delay. We are more interested in the delay of the PHIL-switching process because we can eliminate the overhead of the divert process by implementing the PHIL-switching into the kernel.

The testing results are shown in Table 3.2. First, the results show that the overhead is mainly caused by the divert process. Second, the results also show that the overhead of the divert process and the PHIL-switching process is data-size dependent. If the data-size increases, the overhead becomes worse. For example, the overhead of the PHIL-switching in the test case with 64 bytes is 1,382 microseconds. As data size increases to 1,024 bytes, the overhead is 24,516 microseconds. It is noted that the overhead of divert-socket is worse than the overhead of the PHIL-switching. The reason is that divert socket uses divert mechanism, which is the lower priority, to re-inject packets. Therefore, this could potentially introduce more delay as the packet size increases. Furthermore, the results also motivate us to implement the switch module of the PHIL-switching daemon into the kernel to improve the PHIL-switching performance.

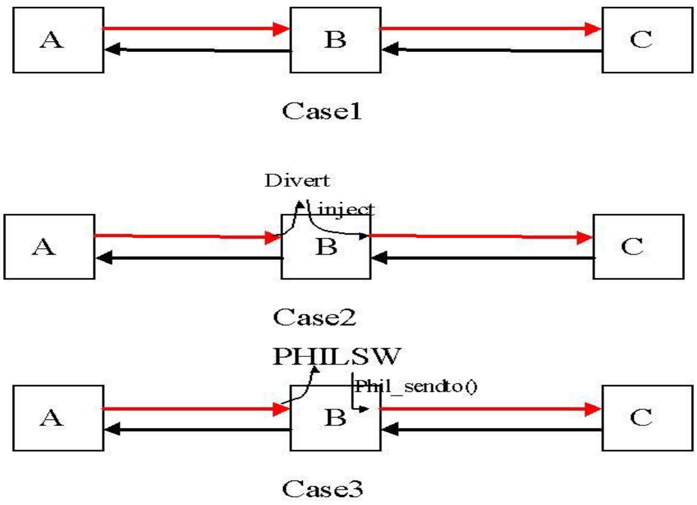


Figure 3. 8 – The test cases for evaluating PHIL- switching

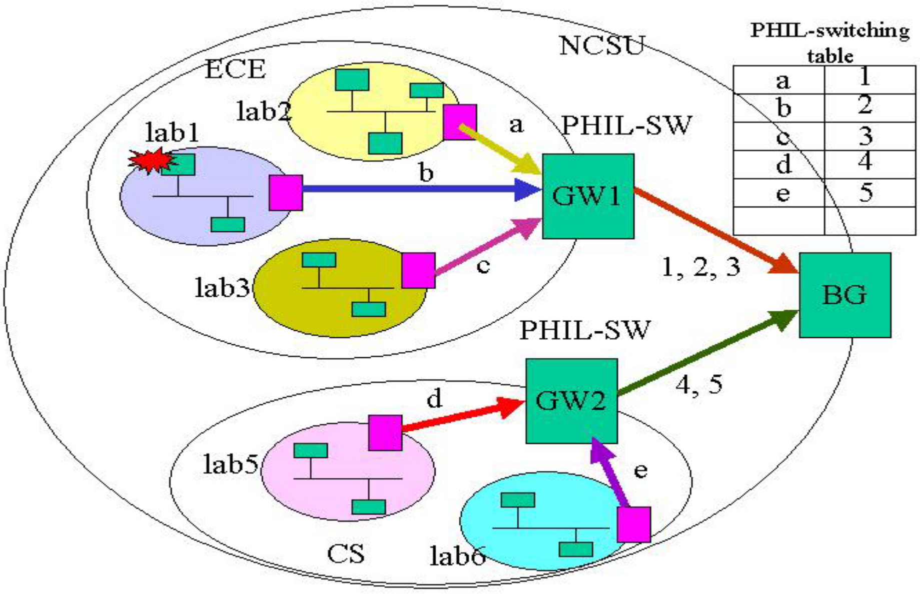
Table 3. 2: The test results of PHIL-switching.

Data size	Case1 (us)	Case2 (us)	Case3 (us)	Divert +divert re-inject overhead	Divert + PHILSW Overhead
16	777	1845	1860	1068 us	1083 us
64	979	2458	2361	1479 us	1382 us
256	1879	19407	13698	17528 us	11819 us
512	2860	39012	27679	36152 us	24819 us
1024	5020	41439	29936	36419 us	24916 us

3.5.4 Applications of PHIL-switching technique

Although the original design of the PHIL-switching is to solve one problem of DECIDUOUS in tracing attacks across different administration domains, the PHIL-switching technique can also be utilized in the local administration domain to improve the attack identification process. The scenario is that the PHIL-switching is used within the local domain to identify the attack flows immediately by pre-setting up the optimal IPSec tunnels. Since the administrators know the entire network topology, they can figure out where to set up tunnels among these routers within the local domain. If there is any trigger for identifying attack flows, the PHIL-switching daemon can automatically activate all tunnels at once-- this is not within the capabilities of DECIDUOUS. If any attack flow is identified, the IDS can just look at the centralized PHIL-switching table and then locate the attack sources immediately.

For example, let us assume there is an administration domain, as shown in Figure 3.9, within NCSU and there are sub-domains in ECE and CS departments. In the NCSU domain, IPSec tunnels have been set up with sub-domains but these tunnels have not been activated so that they will not affect network performance at this moment. If the administrators at NCSU are notified that there is an attack flow from NCSU domain, these tunnels of NCSU domain can be automatically activated to identify the attack flow immediately. If the intrusion detection system in the NCSU domain detects an attack from tunnel 2, the DECIDUOUS can send queries to the centralized PHIL-switching daemon and obtain the attack information which shows the attack is coming from tunnel b (correlated to ECE sub-domain). Then DECIDUOUS passes this attack information to ECE sub-domain. The IDS will keep monitoring any traffic from tunnel b and deactivate other tunnels. In this way, any other innocent traffic will only be slightly impacted during the identification process. After the tunnels were deactivated, the network performance which was briefly adversely affected by these IPSec tunnels will recover right away. After receiving the notification, the ECE sub-domain is able to narrow down the suspected domains to lab1 and finally locate the attackers.



PHIL-switching table

a	1
b	2
c	3
d	4
e	5

Figure 3. 9 – The example of PHIL-switching within the local NCSU domain

3.6 Summary of this chapter

In this chapter, we presented the IPSec-based solution--DECIDUOUS-- to address the problem of IP traceback. However, the DECIDUOUS approach has two distinct disadvantages: first, the IPSec header information, which the intrusion detection system use it to identify attack flows, was dropped after the IPSec process, so DECIDUOUS would not work in most cases. Second, because of the concern of privacy, different administration domains might not want to share their network topologies; thus tracing attack sources across multiple administration domains will pose a problem because DECIDUOUS has no way of knowing where to set up IPSec tunnels.

To overcome these two obstacles, we proposed the Packet Header Information List (PHIL) technique to solve the IPSec-header-dropping problem and the PHIL-switching mechanism to address the cross-domains-tracing issue. The evaluation results show that the PHIL implementation just introduces a small overhead (1 ~ 2 microseconds) while the user-level PHIL-switching introduces much more delay which is actually caused by the divert socket, not PHIL-switching itself. To eliminate the overhead of the PHIL-switching mechanism, pushing the PHIL-switching daemon into the kernel is suggested. With PHIL and PHIL-switching support, we have demonstrated that DECIDUOUS is a feasible and effective solution to the IP traceback problem.

Chapter 4 iTrace-Based Source Tracing

The ICMP traceback (iTrace) scheme is another effective solution to the IP traceback problem. This scheme allows intermediate routers, with low probability (e.g. generating one iTrace packet for an average of 20,000 forwarded packets), to generate an out-of-band ICMP [48] traceback message that is sent along to the destination. “With enough traceback messages (called iTrace packets) from enough routers along the path, the traffic source and path can be determined” [49]. Because the path information is carried on these out-of-band traceback messages, the problem of space limitation (such as the 16-bit IP identifier field limitation in the IP traceback scheme proposed by Savage [6, 8]) does not exist. Therefore, we can put as much path and authentication information as we want into the iTrace packet. For instance, we are able to put the source and destination IP addresses, forward and backward links, and authentication messages into an iTrace packet. These listed path information enable end hosts to figure out attack paths with even a small number (e.g., one) of iTrace packets. Furthermore, because of the out-of-band mechanism, an iTrace packet will not be over-written by other intermediate routers, which makes the authentication problem much simpler. Thus a victim can receive iTrace packets from any intermediate routers equally, no matter how far away the intermediate routers are. Moreover, the ICMP traceback scheme also allows an intermediate router to send iTrace packets toward both destination and source hosts. This option to send iTrace packets toward the source host (called reverse iTrace packets) provides a mechanism to trace the “reflective” DDoS attacks [35] back to not only the reflectors but also the real attack sources. Hence, the iTrace scheme is another effective alternative solution to the IP traceback problem.

4.1 ICMP traceback scheme -- overview

The ICMP Traceback (iTrace) scheme proposed in IETF operates as a daemon running on a router. In addition to forwarding packets, an iTrace-capable router randomly determines if an iTrace message should be sent. This selection must be done probabilistically (and not, with a simple packet counter) [49]. The default probability should be 1 in 20,000, i.e. on average generating an iTrace packet for every 20,000 forwarded packets [49]. The maximum allowed probability (a maximum of 1 in 1,000 is recommended) must not be very high, otherwise there would be excessive iTrace traffic [49].

The iTrace message, sent as an out-of-band ICMP [48, 50] message, contains path information about the router itself (on which interfaces it arrived and left), and previous and next routers [49]. In chosen points of a network, special hosts (called loggers [49]) and program (called analyzer [49]) are designed to log all recent iTrace messages and to extract and then analyze these iTrace messages. For example, the Intrusion Detection System (IDS) can be a logger to collect iTrace packets and can also be a real-time analyzer to re-construct attack paths.

4.1.1 The format of iTrace message

The iTrace message, as shown in Figure 4.1, contains a series of triplets of the form {Tag, Length, Value} (TLV), where the tag is used to identify the type of data stored in the value field, and the length is the size of the variable length of the value field. For example, in Figure 4.1, there is an iTrace message with tag=0x04 (IPv4 address pair), the length of value field is 0x08, which is 8 bytes, and value= (0xc0, 0xa8, 0x45, 0x14, 0xc0, 0xa8, 0x46, 0x2b), which means source address=192.168.69.20 destination address=192.168.70.43. Certain tags may also contain nested TLV's as part of their value fields and there is no limit to the depth of the nesting. Table 4.1 lists the current tags as defined in the latest IETF iTrace draft [17, 49].

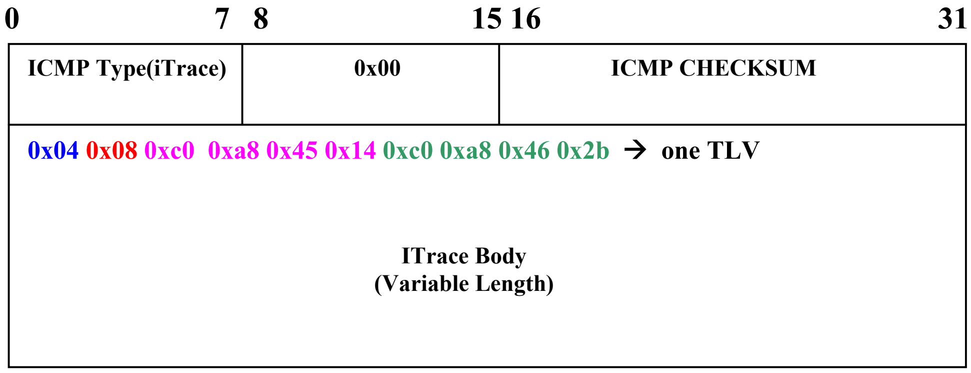


Figure 4. 1 – The iTrace Message Format

Each iTrace message always contains back link and forward link description elements. These two elements describe the links (as defined by a pair of IPv4 or IPv6 addresses), an MAC address pair, and an interface name. Each element in a back or forward link is a TLV. The interface name is the name of the particular interface through which the selected packet was received or forwarded, depending on whether a backward or forward link is being described. For any TLV of the address pair, an upstream address is always specified first, followed by the downstream address. The operator-defined link identifier element, which can be reserved for future use, is one of a set of unique operator-defined names for each link to or from a router. Each iTrace message also contains a timestamp in 64-bit NTP [51] format, where the first 32-bit specifies the number of whole seconds since Jan 1, 1970 00:00:00.00 and the second 32-bit specifies the fractions of a whole second since the last epoch. The tag of the traced packet contents is a portion of the selected packet of at least 32 bytes. The inverse of the iTrace probability is stored for informational purposes. The router identification field contains this router's identifier so the victim can tell whether this router is involved in the attack path(s). (Note that the identifier for each router in the identification field is unique.) Lastly, the remaining tags are used for authenticating iTrace messages to prevent malicious users from sending fake iTrace messages to the victim.

Table 4. 1: List and description of iTrace message tags

Tag	***Element Name***
0x01	Back link description
0x02	Forward link description
0x03	Interface name
0x04	IPv4 address pair describing a link
0x05	IPv6 address pair describing a link
0x06	MAC address pair describing a link
0x07	Operator-defined link identifier
0x08	Timestamp in 64-bit NTP time format
0x09	Traced packet contents
0x0a	Inverse of iTrace probability 1/P
0x0b	Router identification field
0x0c	HMAC authentication data
0x0d	Key disclosure list
0x0e	Key disclosure
0x0f	Public-Key information

4.1.2 The authentication mechanism of iTrace message

To address the issue of fake iTrace messages, the iTrace scheme includes hooks to authenticate iTrace messages by both message authentication codes (MACs) [52, 53] and public key (PK) systems [54, 55, 56, 57]. The iTrace daemon can use its private key to sign each iTrace message. When a logger [49] receives an iTrace message, the signature of this received packet can be verified by the sender's public key. Thus any unauthenticated fake iTrace packet will be dropped. Therefore, the authentication system can effectively solve the problem of fake iTrace packets. However, the authentication process introduces computing overhead, and the key management system [55] becomes more complicated as the Internet continually expands. To reduce the authentication overhead and simplify the key management issue, we propose an effective light-weight authentication scheme to provide a feasible authentication solution for iTrace. The light-weight authentication scheme is discussed in the next chapter.

4.1.3 The operation of ICMP traceback

In an iTrace-capable router, a random determination mechanism is designed to “itrace” packets at random (see Figure 4.2). On average, a random number N (0<N<***P***) is generated in every ***P*** packets, where 1***/P i***s the probability of generating an iTrace message and 1/20,000 (***P***=20,000) is recommended as the default probability in IETF’s ITRACE draft [49]. Based on this random number N, a packet within ***P*** packets will be randomly determined whether it should be “itraced”.

As a packet is selected to be itraced, instead of only forwarding this selected packet, the iTrace-capable router also (1) copies its source and destination IP addresses, (2) combines with the related identification information (backward/forward links, router identifier, and so forth), and then (3) generates an iTrace message for the destination machine or for both source and destination machines, depending on the iTrace mode (forward or reverse iTrace mode).

When the iTrace message is sent, the time-to-live (TTL) field within the IP header [58] is initialized to 255. Each router on the path, regardless of whether they support iTrace, will decrease the TTL field, allowing the victim or receiver of the iTrace message to obtain the relative distance between a particular router and the victim.

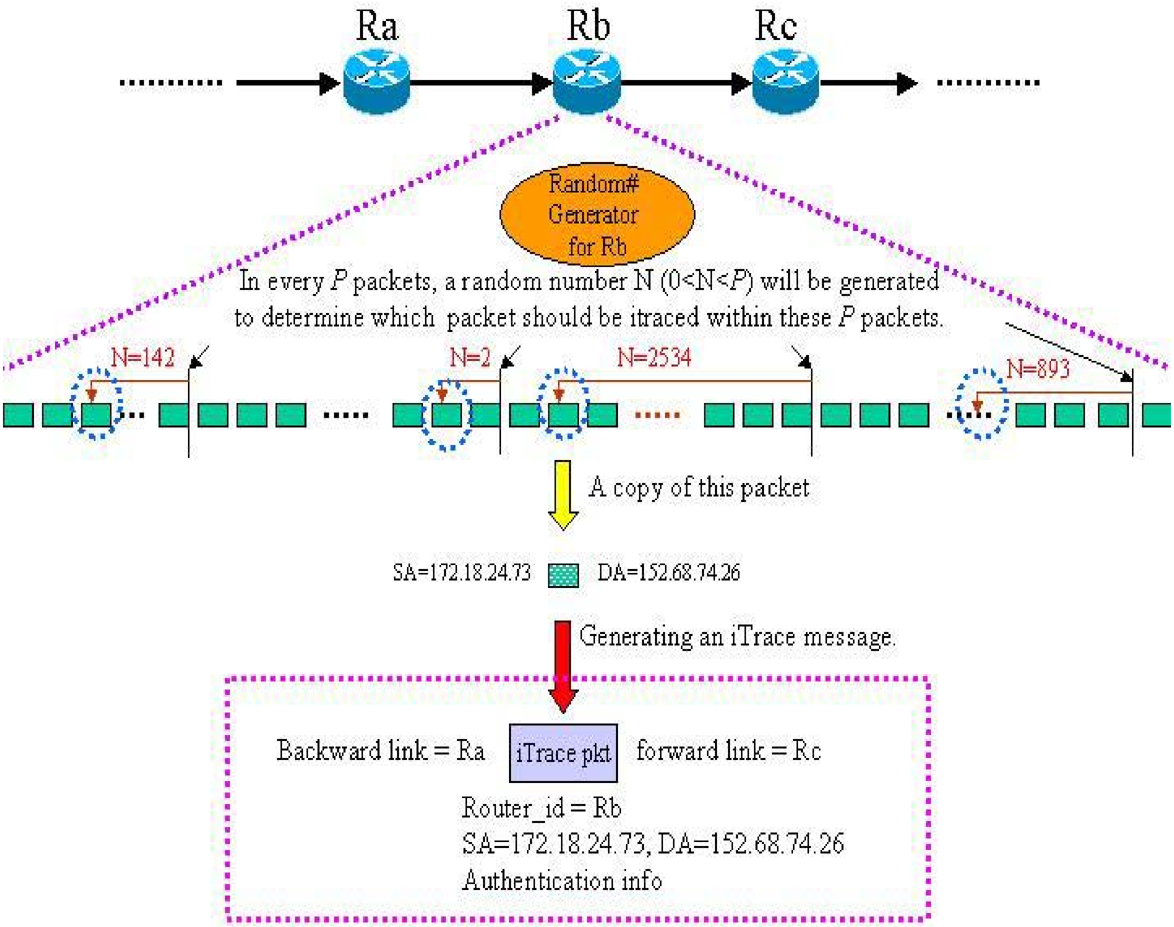
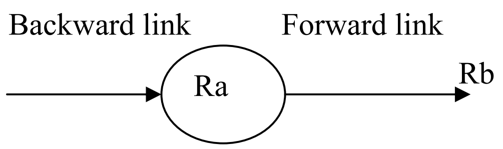
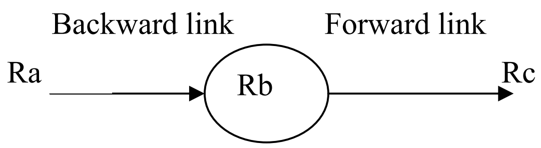
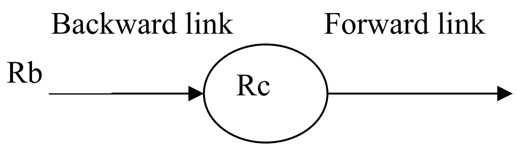

Ra
Rb
Rc
Random# Generator for Rb
In every P packets, a random number N (0<N<P) will be generated to determine which packet should be itraced within these P packets.
N=142
N=2
N=2534
N=893
A copy of this packet
SA=172.18.24.73
DA=152.68.74.26
Generating an iTrace message.
Backward link = Ra
iTrace pkt
forward link = Rc
Router_id = Rb
SA=172.18.24.73, DA=152.68.74.26
Authentication info

Figure 4. 2 – The process of generating an iTrace packet.

After iTrace-capable routers send out iTrace packets, a logger (iTrace-packet collector and analyzer in a victim domain) can collect and analyze these iTrace packets which flow into this victim domain, and ultimately figure out a partial or complete attack path(s). For example, in Figure 4.3 (it is possible for router Ra, Rb, and Rc to “catch” attack packets and then generate iTrace packets for the victim domain) when logger receives iTrace packets from router Ra, Rb, and Rc, it is able to analyze them as follows:







Thus, the logger is able to know that Ra—Rb—Rc is a partial attack path. If the logger puts more partial attack paths together, it is possible to figure out a complete attack path. This complete attack path will help us to trace attack sources and then stop attacks.

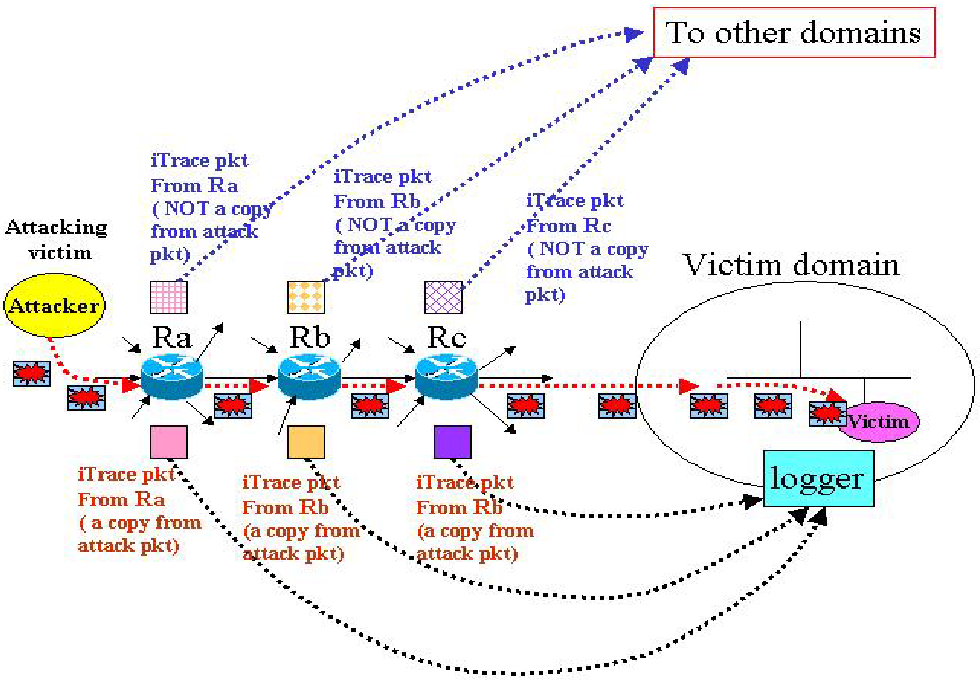


Figure 4. 3 – iTrace packets generated by routers and collected by a logger.

4.2 Potential problems of the ICMP traceback

As a DoS/DDoS attack occurs, people always want to ask the question "How fast can we trace back and stop this attack?" To answer this question, we ask: "how fast can an iTrace-capable router generate a useful iTrace packet for a victim domain?"---that is where the bottleneck occurs. The iTrace scheme deals with "HOW TO" trace back attack sources; however, it hasn't addressed the "how fast" issue yet. For example, (let us assume that the probability of iTrace scheme is 1/20,000) in a high throughput router (10Mpps, packet-per-sec) it takes 1/500 second to generate an iTrace packet; but in a low throughput router (1Kpps), it might take 20 seconds to generate an iTrace packet. Since the router randomly determines which packet to be itraced, it might generate the first "useful" iTrace packet, for instance, at the 200th iTrace process. (Please note that an iTrace packet sent for a victim domain is called a useful iTrace packet, since it is really useful for tracing attack sources.) Thus, this low-throughput itrace-capable router takes about 66 minutes (about one hour) to generate an iTrace packet for a victim domain. In other words, it wastes 199 iTrace resources to generate iTrace packets for the domains which don't desire to receive iTrace packets (not under DoS/DDoS attack or even under DoS/DDoS attack but just don't care). Therefore, how to accelerate the "useful" iTrace packets' generation is a big challenge facing the original iTrace scheme.

4.3 Solution: Intention-Driven iTrace scheme

To solve the problem of the original iTrace scheme, there are two possible approaches: one is to increase the probability of generating iTrace packets. For example, instead of generating an iTrace packet with the probability 1/20,000, we can increase the probability to be 1/10,000 (i.e. on average, generating an iTrace packet for every 10,000 forwarded packets). This approach will definitely accelerate the iTrace packets' generation, especially for low throughput routers. However, there exists a side effect: this approach also increases the net traffic of the Internet. But if we can dynamically adjust the probability, (for instance, increasing the probability when networks are under attacks and setting it back to "normal" probability if networks are not under attacks.) it would be greatly helpful in tracing attacks in low throughput routers. We will discuss this issue in the hybrid model of the iTrace scheme.

Another approach is to boost up the possibility of generating "useful" iTrace packets, even without the need to adjust the probability (for instance, ***P*** is still 1/20,000). In order to design this boost up mechanism, we propose the Intention-Driven iTrace [18, 59] scheme—an enhanced version of the original iTrace scheme. Instead of randomly determining a packet and then directly generating an iTrace packet from those selected packets, the Intention-Driven iTrace scheme uses the same random algorithm to determine/select a packet. However, this determination only generates an iTrace trigger (iTrace Execution-bit) for one of the entries with the desire (this desire can be interpreted by marking the Intention-bit in the entry of the routing table) to receive iTrace packets. In this way, the next coming packet via the entry with the iTrace Execution-bit turned on will be itraced (and the Execution-bit will be reset right after generating this iTrace packet.) For example, in Figure 4.4 after the Intention-bit of routing entry B has been set, the Intention-Driven iTrace scheme will generate iTrace packets only from the entries with the Execution-bit turned on. (Since the domains via routing entry B desires to receive iTrace packets, it means that either these domains are under attacks or they want to be protected from DoS/DDoS attacks). In Figure 4.4, routing entry B is the only possible entry to turn on the Execution-bit, but this rule can apply to multiple entries (multiple attack domains), if their Intention-bits are turned on.

Therefore the Intention-Driven iTrace scheme significantly boosts up the possibility of generating iTrace packets for the domains which really want to receive iTrace packets. For instance, in Figure 4.4, the itrace-capable router can quickly "catch" attack packets and generate useful iTrace packets for the domain (entry B) which is under attack. As a result, this scheme significantly accelerates the process of tracing back attack sources. Meanwhile, this scheme also provides a method to efficiently use iTrace resources: only generate iTrace packets for those entries which really desire to receive iTrace packets. (The original iTrace scheme wastes many iTrace resources to generate iTrace packets for those domains which may not be interested in receiving iTrace packets.)

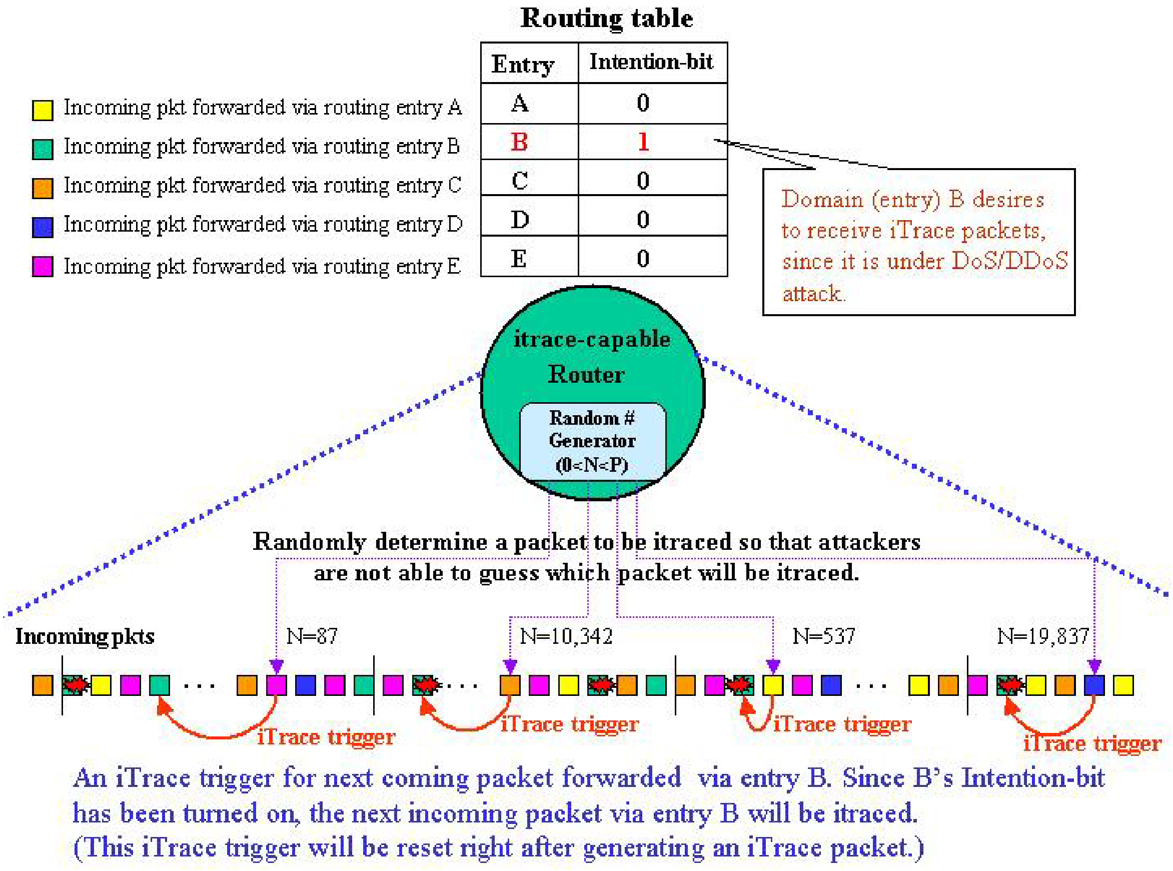

Routing table
Entry | Intention-bit
A | 0
B | 1
C | 0
D | 0
E | 0
Incoming pkt forwarded via routing entry A
Incoming pkt forwarded via routing entry B
Incoming pkt forwarded via routing entry C
Incoming pkt forwarded via routing entry D
Incoming pkt forwarded via routing entry E
Domain (entry) B desires to receive iTrace packets, since it is under DoS/DDoS attack.
itrace-capable Router
Random # Generator (0<N<P)
Randomly determine a packet to be itraced so that attackers are not able to guess which packet will be itraced.
Incoming pkts
N=87
N=10,342
N=537
N=19,837
iTrace trigger
iTrace trigger
iTrace trigger
iTrace trigger
An iTrace trigger for next coming packet forwarded via entry B. Since B's Intention-bit has been turned on, the next incoming packet via entry B will be itraced.
(This iTrace trigger will be reset right after generating an iTrace packet.)

Figure 4. 4 – The Intention-Driven iTrace scheme.

4.3.1 The general architecture of Intention-Driven iTrace scheme

The Intention Driven ICMP Traceback (ID-iTrace) scheme [18, 59] includes three modules: the iTrace scheduler, the iTrace decision module and the iTrace generation module (see Figure 4.5). The iTrace scheduler monitors the incoming traffic and generates iTrace triggers for the decision module as needed. When the decision module receives an iTrace trigger, it first determines which entry in the packet-forwarding table will be the target for the iTrace packet generated by the iTrace daemon, and then it turns on the iTrace Execution-bit of that entry. (Please note that the iTrace Intention-bits belong to the routing table and the iTrace Execution-bits belong to the packet-forwarding table.) The iTrace generation module generates an iTrace packet and then sends this iTrace packet to the destination host. As the forwarding engine forwards a data packet, it will perform a table look-up to find a right entry for this packet. If the iTrace Execution-bit of that forwarding entry has been turned on, a copy of this packet will be sent to the iTrace generation module, and the iTrace Execution-bit will be reset to zero. By controlling these three modules, the Intention Driven iTrace scheme is able to dedicate more iTrace resources to these entries in which the Intention-bits have been turned on. It is noted that the desire of the iTrace messages can be shown by the Intention-bit of that particular entry. For instance, a victim definitely expects to receive iTrace messages for tracing attacks. All routers will respond to the desire of a victim by turning on the Intention-bit for the particular routing entry which is related to the victim domain. By adding the Intention-bit to the routing table, we are able to boost up the probability of "itracing" attack packets for the domains which desire to receive iTrace messages. Moreover, the iTrace-Driven scheme also provides a mechanism to manage the iTrace resources efficiently.

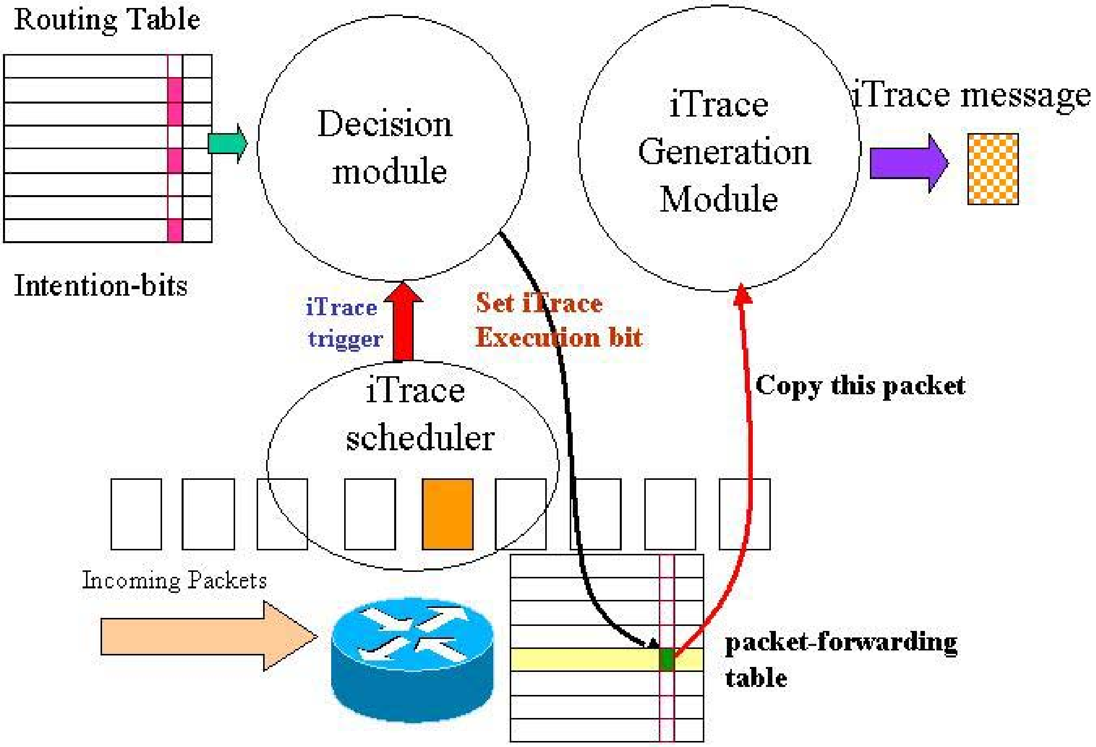


Figure 4. 5 – Decision and iTrace Generation modules of the ID-iTrace scheme.

4.3.2 The design of the Intention-Driven iTrace scheme

The modification of the packet forwarding process is necessary for the Intention-Driven iTrace scheme to work effectively. In addition to forwarding a packet, the modified packet-forwarding process has to check the Intention-bit as well. If the iTrace Execution-bit has been turned on, this forwarding process will also copy this packet for the iTrace generation module to generate an iTrace packet. The pseudo-code for the modified forwarding process is shown as follows:

Modified packet-forwarding process.

```
int
perDataPacketProcess
(IP *p)
{
    ForwardEntry *fe = findForwardEntry(p->destinationIPaddr);
    regularPacketForwarding(p, fe); /* forward this packet first to reduce
                                        the process delay. */

   if (fe == NULL) return -1;
    if (fe->iTrGenBit == 1)
    {
            sendiTrace(p);
            fe->iTrGenBit = 0;
    }

    return 0;
}
```

For the decision module, after a router on average forwarded a packet for every ***P*** packets (***P*** is a new parameter of network management and can be controlled by system administrators), the iTrace scheduler will emit an "iTrace trigger" to the decision module to indicate the need for generating an iTrace message. After receiving this trigger, the decision module has to make a decision: in which entry in the packet-forwarding table should the iTrace Execution-bit be turned on.

After the decision module decides which forwarding entry should be the next iTrace target, the corresponding iTrace Execution-bit will be turned on. If none of the entries has the desire to receive the iTrace messages, then no iTrace message will ever be sent. Hence, the Intention Driven iTrace scheme does not introduce any more iTrace messages than the original proposal does. The following is the pseudo code for the decision module.

The Decision Algorithm to set iTrace Execution-bit

```
int
periTraceTriggerProcess
(RouteEntry *RETable, int RETableSize)
{
    int i;
    RouteEntry *re;
    for(i = 0; i < RETableSize; i++)
       {
             re = &(RETable[i]);
             if (re->intention == 1)
             #ifdef   INTENTION_WITH_DISTANCE
                      enterSelection(re, re->ASPathLength);
             #else
                      enterSelection(re, 1);
             #endif   INTENTION_WITH_DISTANCE
                      }

              re = finalSelection();
     if (re != NULL)
    re->fe->iTrGenBit = 1;
              return 0;
}
```

4.3.3 Main concerns about the Intention-Driven iTrace design

There are two main concerns about the Intention-Driven iTrace design. The first one is iTrace resource adjustment. In an ideal situation, when there are no attacks, the iTrace daemon may dedicate iTrace resources for other purposes (for instance, monitoring network traffic) or even suspend the iTrace process to reduce iTrace traffic. Then when DoS/DDoS attacks occur, the iTrace daemon should be able to make adjustments and dedicate more iTrace resources for tracing attacks. Such an efficient and flexible control mechanism for the iTrace design is lacking in the Intention-Driven iTrace scheme. The second concern is the routers' vulnerability to incorrect configuration and to 'sabotage' by the attackers. If a router is not correctly configured or if the attackers manage to compromise an itrace-capable router by changing its routing table (altering the Intention-bit) such that the iTrace daemon is unable to pick up the attack traffic, then this itrace-capable router is unable to generate useful iTrace messages for the victim domain. Thus the logger (victim or the IDS) has no way of tracing the real attackers. In view of these two concerns, it is necessary to redesign the Intention-Driven iTrace scheme to make it a robust and efficient control mechanism.

4.4 The more efficient solution: the Hybrid iTrace scheme

To address these two issues (robustness and efficiency), we propose a Hybrid solution: integrating both the Intention-Driven iTrace scheme and the original iTrace scheme. First, we design a control mechanism for system administrators to flexibly and dynamically control iTrace resources: dedicating the iTrace resources to other purposes when there are no attacks and adjusting the iTrace resources to trace attackers when DoS/DDoS attacks occur. Second, this solution reduces the vulnerability of routers: if a router is incorrectly configured or is compromised by attackers so that the Intention-bit of the victim prefix is set to zero, this Hybrid approach is still able to itrace the attack traffic from the original iTrace scheme. Therefore, the Hybrid iTrace scheme effectively deals with the shortcomings of both the original iTrace scheme and the Intention-Driven iTrace scheme.

4.4.1 The detailed design of the Hybrid iTrace scheme

For the control mechanism, we introduce a new component-- probability controller. First, it provides interfaces for system administrators to flexibly adjust the probability so that the low-throughput routers have higher probability (e.g. 1/10,000) to accelerate iTrace packets' generation and the high-throughput routers have lower probability (e.g. 1/30,000) to reduce the iTrace traffic. Second, this probability controller also regulates the probability of the iTrace mechanism in either using the Intention-Driven iTrace scheme (efficiently "itracing" attack packets) or the original iTrace scheme (avoiding the issue of mis-configuration or compromise.) The Hybrid iTrace approach can be implemented in two ways: Traffic-Separation mode and Probability-Separation mode. The Traffic-Separation mode is to separate the incoming traffic into two classes: intention traffic and regular traffic. (Intention traffic contains traffic through the entries in which the Intention-bit is set to one. Regular traffic contains traffic through the entries in which the Intention-bit is set to zero.) And then intention traffic will be itraced using the Intention-Driven iTrace scheme, while regular traffic will be itraced using the original iTrace scheme. The Probability-Separation mode regulates the probability of the iTrace generation: let the probability of generating iTrace packets through the Intention-Driven iTrace scheme be ***q*** and that of generating iTrace packets through the original iTrace scheme be (1-***q***). If there are DoS/DDoS attacks, we can simply increase the probability ***q*** to dedicate more iTrace resources to tracing attacks through the Intention-Driven iTrace scheme. We will present both the Traffic-Separation and Probability-Separation modes in the following sections.

4.4.1.1 Hybrid iTrace scheme: Traffic-Separation mode

Figure 4.6 illustrates the design of the Traffic-Separation mode:

First, the probability controller provides interfaces for system administrators to control two things: (1) the expected random-determination probability ***1/P*** (randomly "pick" a packet on average for every ***P*** packets) and (2) the controlled probability ***q*** ***(***dedicating iTrace resources in the Intention-Driven scheme). The ***Ratio_intention*** is an initial value to tell the probability controller what the current ratio of intention traffic is. This initial value can be obtained from network management (SNMP), or the controller can monitor the incoming traffic and then figure out this value. The above three parameters (***1/P***, ***q***, and the initial value ***Ratio_intention***) can be converted by the probability controller into ***Rand#_I*** (the converted random-determination probability for intention traffic is ***1/Rand#_I***) and ***Rand#_R*** (the converted random-determination probability for regular traffic is ***1/Rand#_R***.)

Second, the incoming traffic will be separated into two classes: intention-traffic (Intention-bit equals to one) and regular traffic (Intention-bit is zero). Intention-traffic will be itraced through the Intention-Driven iTrace process based on the random-determination probability ***1/Rand#_I*** (on average for every ***Rand#_I*** packets), while regular-traffic will be itraced through the original iTrace process based on the random-determination probability ***1/Rand#_R*** (on average for every ***Rand#_R*** packets.)

By controlling the values of ***1/P*** (random-determination probability***)*** and ***q*** (the controlled probability in either the Intention-Driven iTrace process or the original iTrace process), we can adjust ***Rand#_I*** and ***Rand_R*** to control iTrace resources. For example, in the case of DoS/DDoS attacks, we can decrease ***Rand#_I*** so that most iTrace packets will be generated through the Intention-Driven iTrace process. As a result, we can dedicate most iTrace resources to tracing attacks. On the other hand, when there is no attack, system administrators can decrease ***Rand#_R*** so that most iTrace packets will be generated through the original iTrace scheme. Therefore, the Traffic-Separation mode of the Hybrid iTrace scheme can effectively control iTrace resources by adjusting the random-determination probability ***1/P*** to satisfy different needs.

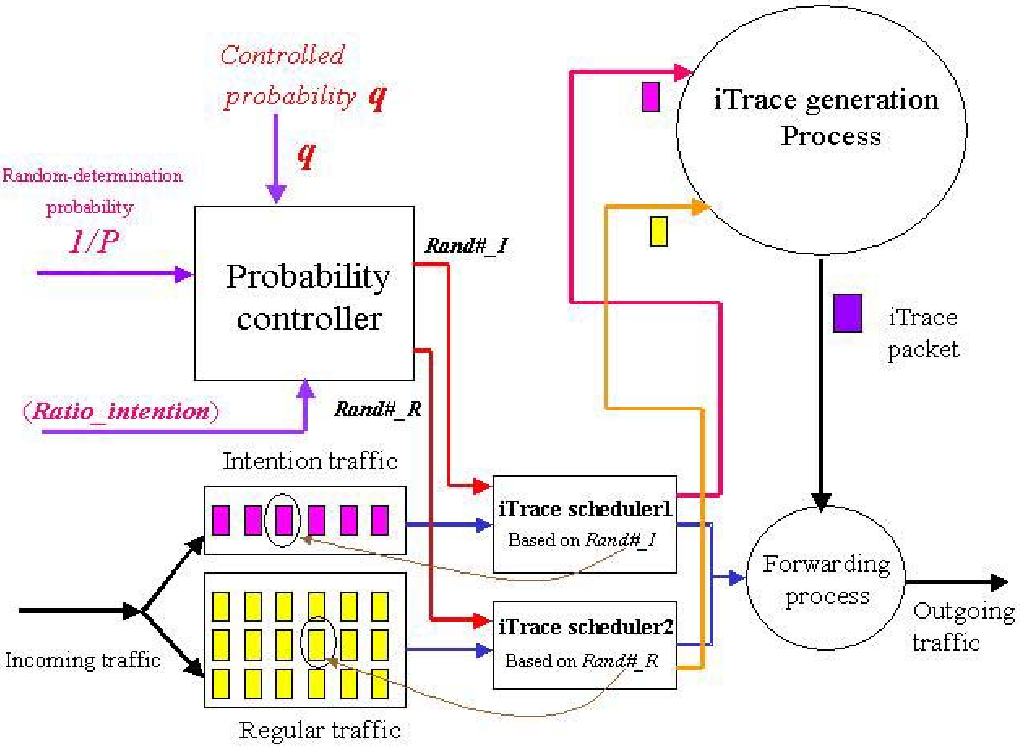


Figure 4. 6 – The Hybrid iTrace scheme: Traffic-Separation mode

The pseudo code for Traffic-Separation mode is as follows:

The algorithm for generating iTrace packets in the Traffic-Separation mode.

```
int
perDataPacketProcess
(IP *p)
{
    ForwardEntry *fe = findForwardEntry(p->destinationIPaddr);
    regularPacketForwarding(p, fe); /* forward this packet first to reduce
                                       the process delay. */
    if (p->IntentionBit)
            IntentionDriveniTraceScheduler(p); /* Intention traffic class */
    else
            OriginaliTraceScheduler(p); /* Regular traffic class. */

    return 0;
}
```

4.4.1.2 The Hybrid iTrace scheme: Probability-Separation mode

The Probability-Separation mode is shown in Figure 4.7. The iTrace scheduler will capture a packet (no matter what type of traffic it is) on average for every ***P*** packets. This captured packet will be sent to the probability controller. Then the probability controller will decide whether it should go through the Intention-Driven iTrace process (in a probability ***q***) or the original iTrace process (in a probability (1- ***q***)). Please note that this mode allows system administrators to control not only the random-determination probability ***1/P*** but also the controlled probability ***q***. In a random-determination probability ***1/P***, system administrators are able to control the iTrace resources by adjusting the value of ***q***: raising the value of ***q*** will increase the probability of the captured packet going through the Intention-Driven iTrace process and so more iTrace resources are dedicated for these entries which desire iTrace messages. Furthermore, system administrators can also adjust the random-determination probability ***1/P*** to accelerate the capturing process, especially for the low-throughput routers. For example, in case of DoS/DDoS attacks, we can utilize more iTrace resources by adjusting the value of ***q***, and also shorten the time which victim domains need to trace attack sources by adjusting the value of ***1/P***. Therefore, the Probability-Separation mode of the Hybrid iTrace scheme is also able to effectively use iTrace resources by controlling the value of ***q*** and ***1/P***.

The pseudo code of Probability-Separation mode is shown as follows:

The algorithm for generating iTrace packets in the Probability-Separation mode.

```
Int
perDataPacketProcess(IP *p)
{
    ForwardEntry *fe = findForwardEntry(p->destinationIPaddr);
    regularPacketForwarding(p, fe); /* forward this packet first to reduce
                                       process delay. */
    if (Scheduler(p) >0)  /* it is time to trigger iTrace process */
    {
            if (Random#() < IntentionProbability)
                    IntentionDriveniTrace(p); /* go for Intention-Driven iTrace */
            else
                    OriginaliTrace(p); /* go for original iTrace */
     }
    return 0;
}
```

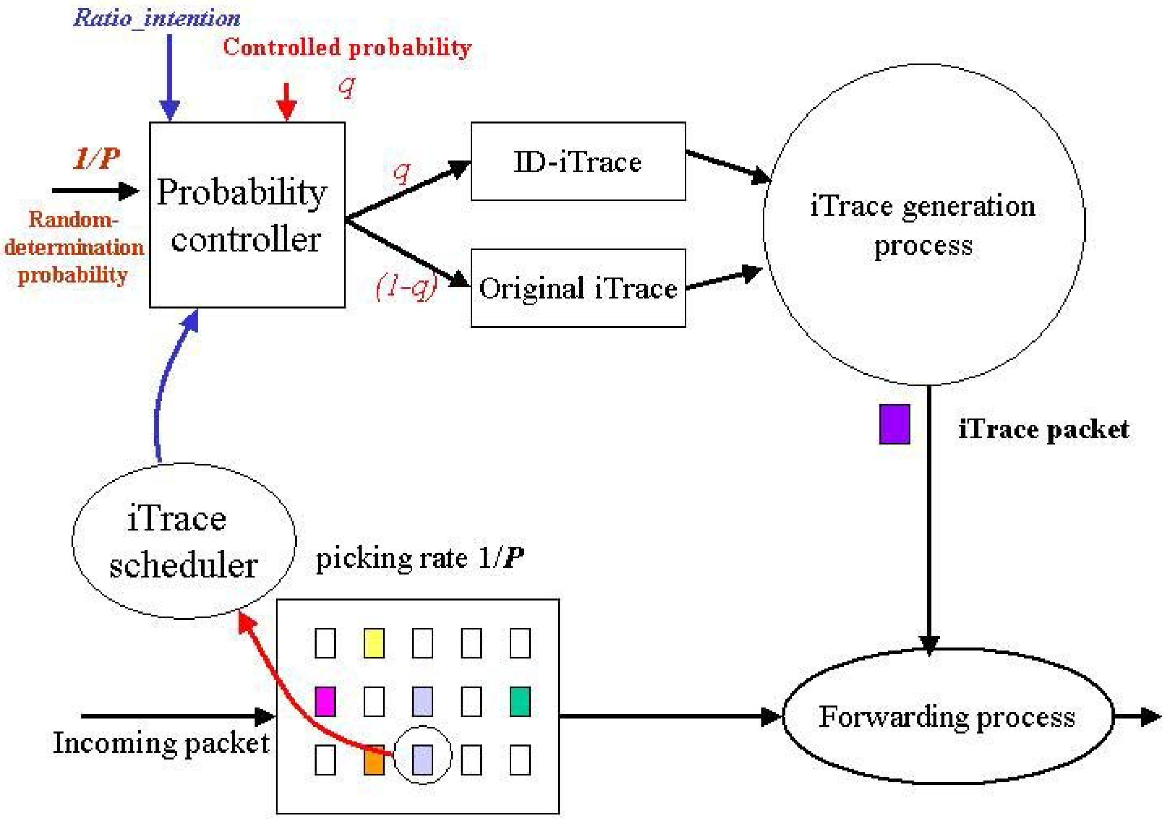


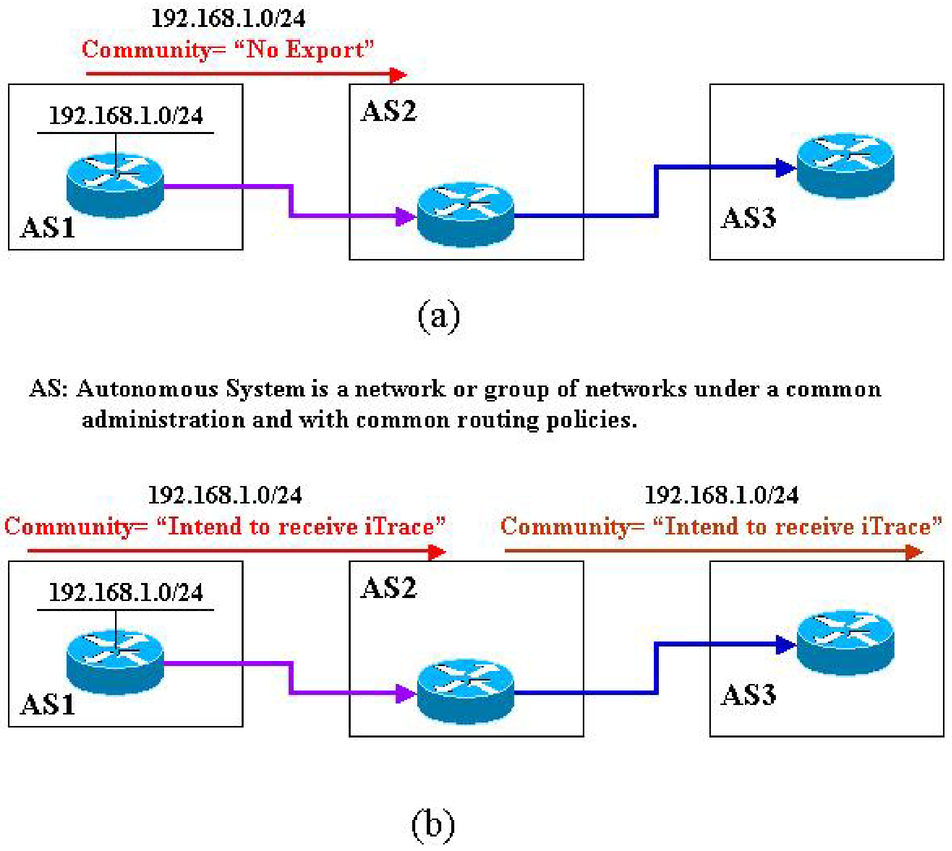
Figure 4. 7 – The Hybrid iTrace scheme: Probability-Separation mode.

4.5 The distribution of the Intention-bit

How to distribute the Intention-bit values is a crucial part of the Intention-Driven iTrace and the Hybrid iTrace schemes in their endeavor to efficiently select attack packets and then generate useful iTrace packets. In this section, we propose a solution to address this issue: using the BGP as a vehicle to distribute the Intention-bit values.

The Border Gateway Protocol (BGP) [60] is a routing information exchange protocol. Routers running BGP can exchange network reachability information with other routers in an effort to make efficient routing decisions. In the case of routing changes, the BGP router of this corresponding domain will advertise an address prefix with associated properties to update the routing information of the downstream BGP routers. These properties are referred to as BGP attributes. BGP defines a variety of route attributes, such as AS_PATH, NEXT_HOP, LOCAL_PREF, MED (Multi-Exist Discriminator), and COMMUNITY. Among these attributes, we are only interested in the community attribute [61, 62], since we can use it to carry the values of the Intention-bit. How the community attribute works can be illustrated by Figure 4.8-(a). AS1 advertises 192.168.1.0 to AS2 with the community "no-export". AS2 will propagate the route throughout AS2 but not send this route to AS3 or any other external AS.

The Intention-bit values are distributed as follows: when a BGP router advertises its reachability to its downstream BGP router(s), it also attaches the Intention-bit value for that particular network address prefix (see Figure 4.8 -(b)). Thus, when this BGP update is propagated to downstream BGP routers, the Intention-bit value of the corresponding route will be updated. Furthermore, if BGP updates are aggregated, the Intention-bit values will also be aggregated.



(a)

AS: Autonomous System is a network or group of networks under a common administration and with common routing policies.

(b)

Figure 4. 8 – (a) "No Export" community (b) Our proposed iTrace community.

Since the community attribute [61, 62] in BGP is utilized to carry the information of the BGP update, we propose adding two new community string values: one for "intend to receive iTrace messages" and the other for "not intend to receive iTrace messages". Furthermore, the community attribute is transitive and optional, which means that if a router does not understand these new community values (for instance, an old BGP router cannot support Intention-bit), it will still forward this BGP update packet to the next hop. Eventually, a downstream BGP router can pick it up and update its intention bit values. Please note that this proposal does not require changes to all the routers, since we do not introduce a new BGP attribute, but only two new values for the existing BGP community attribute. Therefore, when a particular network is under DDoS attacks, the intrusion detection system will inform the corresponding BGP router to include the iTrace intention value in the next BGP update. Because the rate of the Intention-bit updates can be limited by the BGP keepALive interval, this design will not introduce extra BGP traffic while the intention bit value can be distributed through the entire Internet.

4.6 Summary of this chapter

In this chapter, we introduced another IP traceback approach--the ICMP traceback (iTrace); and we also analyzed how this scheme handles both DoS/DDoS and reflective DDoS attacks. (Until today, only iTrace scheme has the capability to handle reflective DDoS attacks.) After investigating this scheme, we pinpointed the original iTrace scheme's problem: this scheme may waste many iTrace resources and a lot of time in generating useless iTrace packets for those end hosts which just ignore them; at the same time it has little or no chance of generating iTrace messages for those hosts which really desire to receive them. To solve this problem, we propose the Intention-Driven iTrace (ID-iTrace) scheme.

In the ID-iTrace scheme, the Intention-bit is introduced to indicate the desire for iTrace message on a router's routing table. (The values of the Intention-bit will be distributed to the entire Internet through the BGP update mechanism, as in our proposal.) For any "iTrace trigger", the decision module of the ID-iTrace scheme will check those entries with Intention-bit turned on and decide which entry should be the one to turn on the Execution-bit. For the next incoming packet with the Execution-bit turned on, the iTrace generation module will not only forward but also itrace the packet, and then reset the entry's Execution-bit.

Although the ID-iTrace is powerful enough to solve the problem of the original iTrace scheme, it has its disadvantages: if a router is incorrectly configured or could be compromised so that the values of the Intention-bit are modified, this scheme is totally unable to generate useful iTrace messages. Furthermore, a flexible control mechanism is required to efficiently control the iTrace resources. Based on these concerns, we propose the

Hybrid iTrace solution: integrating both the Intention-Driven iTrace and the original iTrace schemes to replace the Intention-Driven iTrace design. With the merit of flexible control of iTrace resources, the Hybrid iTrace scheme is able to provide a complete and effective solution to trace DoS/DDoS attacks.

Chapter 5 Light-Weight Authentication for ICMP Traceback

An authentication scheme [49] for iTrace/ID-iTrace is needed to prevent malicious users from sending falsified or spoofed iTrace messages toward the victim to affect the traceback results. An attack targeted on the iTrace mechanism is easy to launch because the iTrace message format is well known and the capability of spoofing the IP datagrams containing the fake iTrace messages is available in most operating systems with only a small amount of RAW socket programming [63]. Therefore, we need an authentication scheme which, on the one hand, would prevent spoofed iTrace messages, and, on the other hand, not put an undue burden on the routers which provide the authentication services.

One proposal to solve the authentication problem is for the original iTrace scheme to adopt a public key system: a central authority issues every router with a private key and a corresponding public key, which is available to the public. The router, from which an iTrace message originated, will sign it using its private key. The receiver of the iTrace messages can verify the authenticity by checking the signature with the corresponding public key of the sending router. If the signature can be verified, the receiver can trust the source of the iTrace messages and deem the message authentic. However, this public key system has key management and cost issues that make this solution impractical. First, there are too many routers on the Internet for a central authority to maintain the public-private key pairs. In addition, a central authority could be a new victim for denial of service attacks. If the central authority was taken down during a denial of service attack, the iTrace messages cannot be verified. Second, public key cryptography operations are expensive: such a public key system would most likely require user-space processing and introduce a context switch into the routing process.

5.1 The light-weight authentication system

To address the above authentication issues, we propose the light-weight authentication approach which is a much simpler mechanism: each iTrace daemon has one private key K1 to pre-compute a 256 *256 digest table, which has 256 entries and each entry has 256 bits (see Figure 5.1). Please note that the pre-computing process only executes once, aiming at reducing the computing cost.

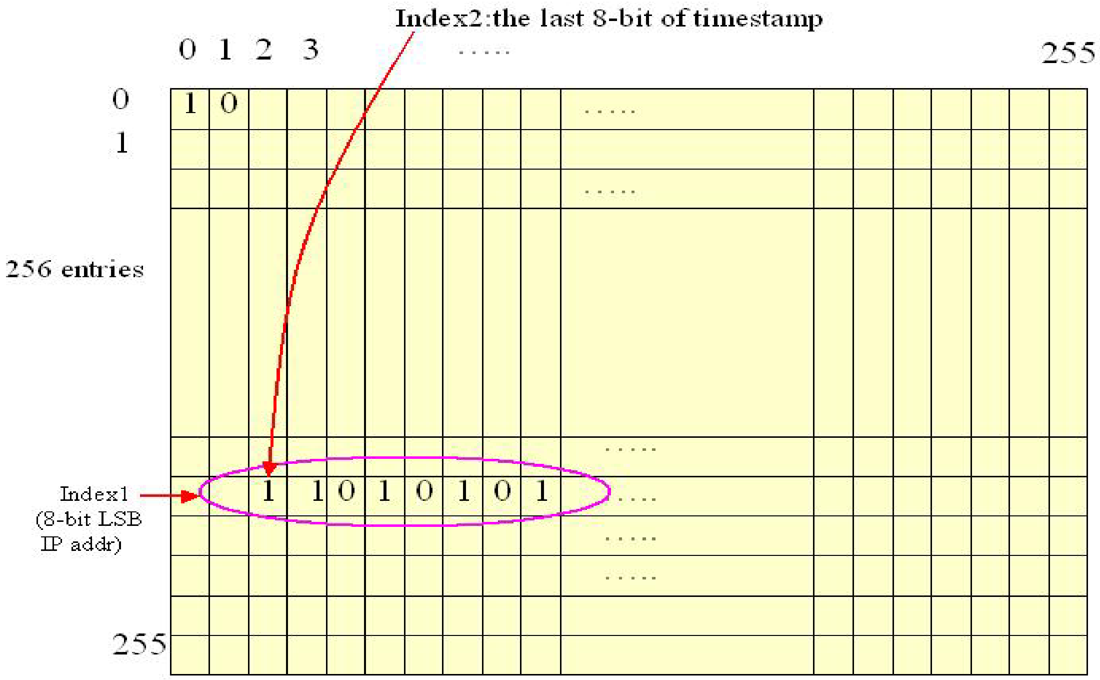


Figure 5. 1 – The pre-computed digest table.

When a router generates an iTrace message, the selector, as shown in Figure 5.2, uses the last significant 8 bits of the destination address and the last significant 8 bits of the time stamp as indexes to obtain 8-bit digest from the pre-computed authentication table. This 8-bit digest will combine with the last significant 8 bits of the time stamp to form a 16-bit authentication information. This information will be put into the identifier field of IP header and then will be sent toward the destination (or victim).

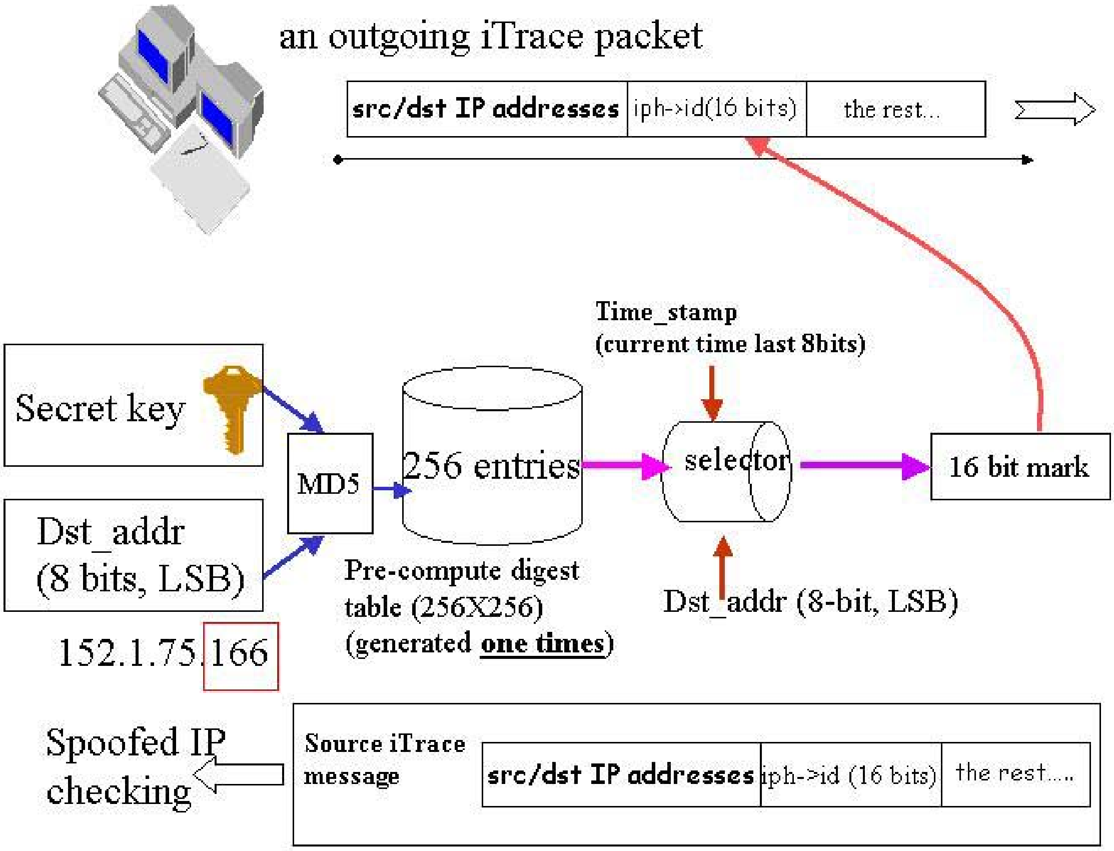


Figure 5. 2 – The light-weight authentication design for iTrace

When a victim or a host receives iTrace messages, it can verify all the iTrace messages or select a small subset at random to verify. (The latter helps to reduce network overhead introduced by the authentication scheme.) During the verification, the receiving host sends an authentication request (see Figure 5.3) to the router from which the iTrace message originated. This request includes the original iTrace message, the light-weight authentication information, and a random number ***R***, which is generated by the receiving host. This router can verify the authenticity by using the source (i.e. the end host's) IP address and the most significant 8 bits of the authentication information (16 bits) as indexes to obtain an 8-bit digest. By comparing this 8-bit digest to the last significant 8 bits received, this router is able to verify whether the iTrace packet was the one it originally generated. If the iTrace message was issued from this router, the two 8-bit digests should be identical; then the iTrace message is deemed authentic. The router will reply to the host, which requests authentication, with ACK and ***R+1***. Otherwise, it will send back a predefined failure message (see Figure 5.3)

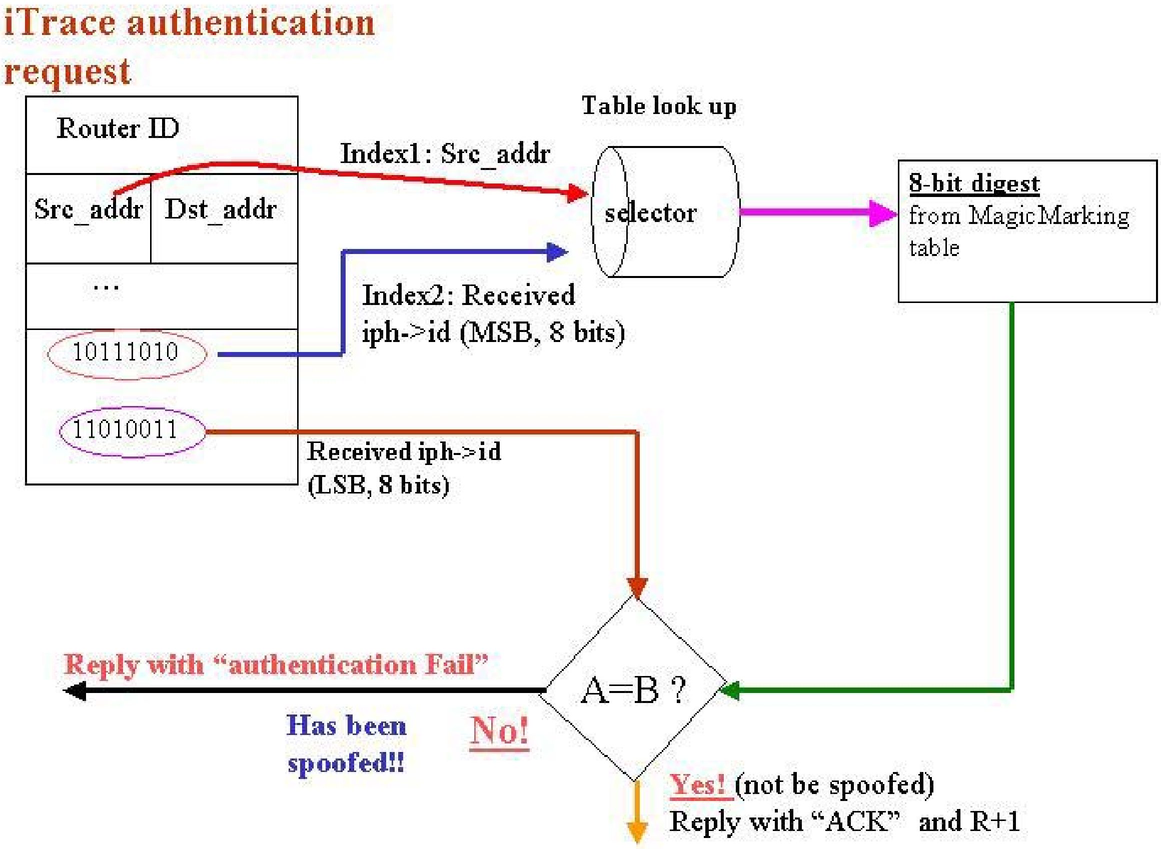

iTrace authentication request
Router ID
Src_addr
Dst_addr
…
10111010
11010011
Table look up
Index1: Src_addr
selector
8-bit digest from MagicMarking table
Index2: Received iph->id (MSB, 8 bits)
Received iph->id (LSB, 8 bits)
Reply with "authentication Fail"
A=B ?
Has been spoofed!!
No!
Yes! (not be spoofed)
Reply with "ACK" and R+1

Figure 5. 3 – The verification process of an authentication request from victims.

5.2 The evaluation of the light-weight authentication scheme

This authentication mechanism is aimed at solving the key management problem and reducing the cost of the authentication operation. In this section, we evaluate the overhead of this scheme in detail.

The light-weight authentication scheme is implemented in Linux kernel (version 2.0.36) with Pentium II 450 machines. For evaluating the overhead, 100000 TCP echo request packets are issued. The echo reply packets are collected to measure the average delay. Please note that since the overhead is so small (less than 9 microseconds), any other processes might affect the accuracy of measuring the maximum overhead. Thus in this testing we evaluate only the average overhead.

Table 5. 1: The overhead of the light-weight authentication scheme

Data size	Without Auth.	With Auth	Overhead (average)
6	755 us	762 us	7 us
12	1002 us	1005 us	2 us
25	1498 us	1505 us	7 us
51	2492 us	2498 us	6 us
102	4476 us	4448 us	9 us

In Table 5.1, the overhead is calculated by subtracting the average time of the authentication process from the average time without the authentication process. The test results show that the average overhead is about 9 microseconds. It turns out that the overhead of this authentication scheme is small and not significant for most applications.

5.3 Summary of this chapter

The light-weight authentication design not only removes the key management issue but also avoids the cryptography operation by pre-computing the digest table (the only cost is table lookup). From our evaluation, we have shown that this scheme can effectively prevent malicious users from forging iTrace messages, and only costs a small amount of overhead.

Chapter 6 Simulation-Based Evaluation

One of the objectives of this exercise is to simulate the original iTrace problems: if the attack rate is low, (For example, DDoS attackers can compromise thousand of machines and control them to send attack packets in low packet rate in each compromised machine; however, these low-rate attack packets finally will be aggregated and then be used to “flood” the target), there is little or no chance for the original iTrace scheme to generate useful iTrace messages. Furthermore, even if this scheme has a small chance of generating a useful iTrace message for victims, it will take a long time to generate one useful message since the original iTrace scheme wastes a lot of iTrace resources in generating useless iTrace messages.

To address these fundamental issues, we propose the Intention-Driven iTrace scheme and then the Hybrid iTrace scheme. Therefore, another objective of this simulation is to compare their capabilities in generating useful iTrace messages – not only in quantity (the number of useful iTrace messages) but also in quality (how fast they can generate useful iTrace).

In this exercise, we have two strategies: first, we focus on simulating the problems of the original iTrace scheme and then compare its capability of generating useful iTrace packets with the Intention-Driven iTrace scheme’s capability. Second, we focus on the comparisons among the Original iTrace, Intention-Driven iTrace, and Hybrid iTrace schemes. We also verify the control ability of the Hybrid iTrace scheme to affirm its design requirements.

6.1 The simulation environment

In our simulation, we adopt Network Simulator Version2 (NS2) [64, 65, 66] to generate normal traffic and attack traffic. On NS2, we create an iTrace package to support original iTrace, Intention-Driven iTrace, and Hybrid iTrace schemes. The iTrace functions are implemented in two files: itracerouter.cc and classifier.cc. The attack package generating attack traffic is also implemented in the file of ping_attack.cc. The details of NS2, the iTrace package and the attack package can be found in [67]

The simulation test bed with 128 nodes is shown in Figure 6.1. We choose node-125 and node-41 as victims. Node-21, 25, 87, 93, and 121 are attack sources to launch DDoS-like attack. Other nodes are used to generate regular traffic so that the relative attack rate is small. Thus we can simulate the issues of the original iTrace scheme: the original iTrace scheme has little chance or no chance to itrace a useful iTrace message if the attack rate is small.

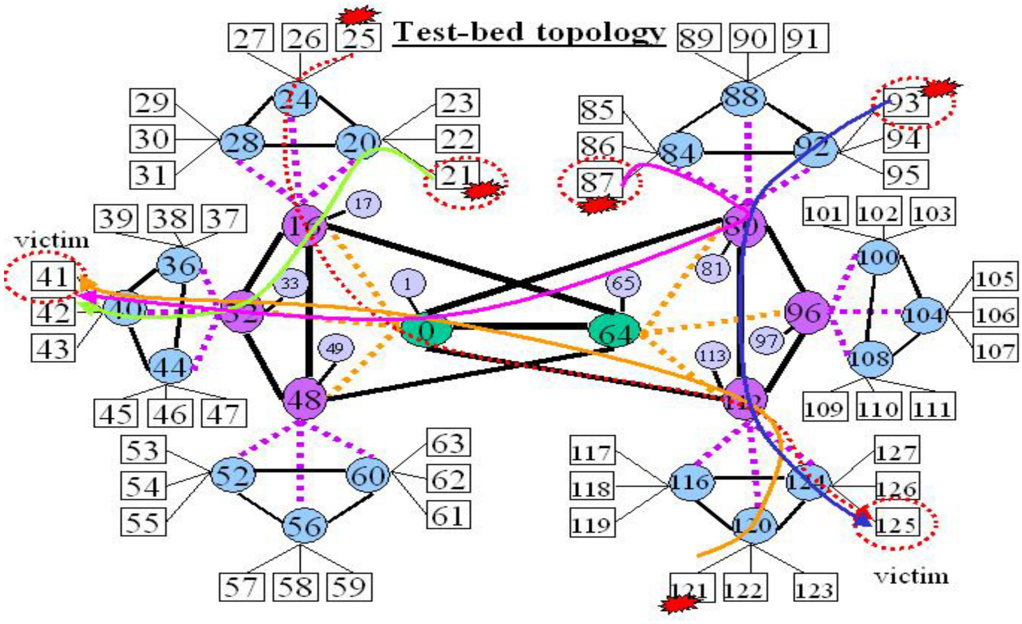


Figure 6. 1 – The simulation test-bed

6.2 The measurements of this simulation

An iTrace daemon is expected to generate iTrace packets for end hosts or victims who really desire to receive iTrace packets as soon as possible. In order to measure this "expectation", we introduce the concept of "Usefulness" and "Value":

- Usefulness

  If the packet contained in an iTrace message is an attack packet and the node is interested in receiving iTrace messages, this iTrace message is "useful".

- Value

  The router information value of an iTrace message (iTr) is determined by the following five parameters:

  Value(iTr) = F(Attack[iTr.pkt], Intention[iTr.dst-ID], HopCount[iTr.rtr-ID->iTr.dst-ID], Received[iTr.rtr-ID->iTr.dst-ID], Generated[iTr.rtr-ID]),

  where Attack[iTr.pkt] indicates whether the itraced packet is an attack packet, the Intention[iTr.dst-ID] is the Intention-bit, HopCount[iTr.rtr-ID-> iTr.dst-ID] represents the distance (HopCount) from the router to the destination (victim), Received[iTr.rtr-ID->iTr.dst-ID] represents how many iTrace messages from iTr.rtr-ID to iTr.dst-ID have been received before, and Generated[iTr.rtr-ID] represents the number of iTrace messages already generated by iTr.rtr-ID for all destinations. Here we assume that any iTrace message generated by a router will be received by victims. Please note that, while the function "F" can be defined in many different formats, in this dissertation, a simple form of "F" is defined as follows:

Value(iTr) =

(Attack[iTr.pkt] * Intention[iTr.dst-ID] * HopCount[iTr.rtr-ID->iTr.dst-ID])

/ ((Received[iTr.rtr-ID->iTr.dst-ID] + 1) * (Generated[iTr.rtr-ID] + 1)),

The simple form of “value” can be explained as follows.

1. If the packet is not the attack packet or the destination is not interested in receiving iTrace (Intention-bit=0), then the vale of this iTrace messages is 0 (valueless).
2. If the router is far away from the victim, it will get more “value”. In other words, we are more interested in iTrace messages which are generated by the router near the attack sources.
3. ((Received[iTr.rtr-ID->iTr.dst-ID] + 1) * (Generated[iTr.rtr-ID] + 1)) implies that the duplicated iTrace message has less value.

6.3 Strategy I: the comparison between the original iTrace and the ID-iTrace schemes

In the first strategy, we focus on the comparison of the capability of generating useful and valuable iTrace packets. There are two scenarios in our simulation: single-attack vs. single-victim and multiple-attacks vs. multiple-victims. The path of one single attack is 25->24->16->0->112->124. There are five attack paths in the case of multiple-attacks vs. multiple-victims:

25->24->16->0->112->124->125

93->92->80->112->124->125

21->20->16->->32-40->41

87->80->0->32->40->41

121->120->112->0->32->40->41

We generate regular traffic and attack traffic in our simulation test bed and then run the simulation to compare the capabilities of the original iTrace scheme and the Intention-Driven iTrace scheme.

6.3.1 Simulation results and analysis

One of the attack sources is node25, as shown in Figure 6.2. On router 24, which is near the true attack source (node25), the simulation results are obtained as follows:

node 24's itraced_table   Total itraced pkts= 1200

Src	Dst	count	%	intention	# of usefulness
25	125	5	0.004167	1	5
126	26	1195	0.995833	0	0

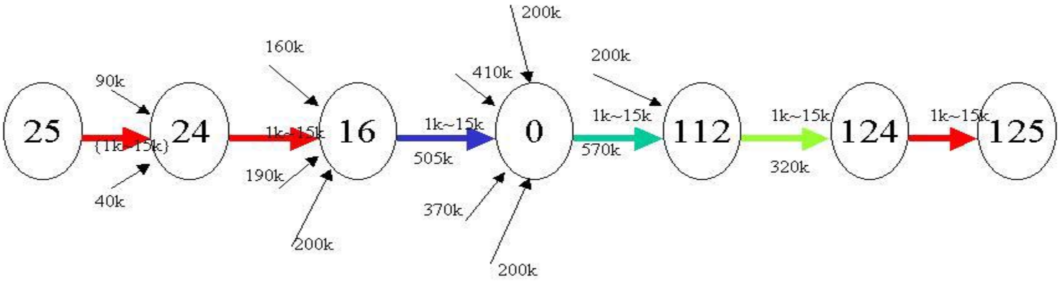


Figure 6. 2 – An attack path

It is not surprising that, because of the amount of background traffic, for the first 1200 iTrace messages generated by router 24, most (99.58%) of them were not toward the victim. After examining closely the simulation log, we found that the generation sequence numbers of those five useful iTrace messages by router 24 were #293, #429, #707, #817, and #1107. i.e., the first useful iTrace message was generated after router 24 generated 292 useless iTrace messages towards other destinations. If one iTrace message represents 20,000 data packets, this implies that the first useful iTrace message is generated after router 24 forwarded about 6 million packets.

The measurement of "usefulness" does not show an important aspect of iTrace behavior: how fast the router generates useful iTrace messages. In responding to DDoS attacks, in order for a system administrator to quickly recover from the damage, it is highly desirable to receive iTrace messages immediately after the attack. For instance, in this example the first iTrace message from router 24 towards the true victim 125 is generated after router 24 generated 292 useless iTrace messages (roughly router 24 has forwarded 6 million packets). If router 24's throughput is 10,000 packets per second, it will take about 600 seconds before router 24 is identified. Clearly, it will be better if we can receive this first iTrace message at an earlier stage.

According to the definition of value, most of the first 1200 iTrace messages generated by router 24 in this example are valueless except:

Value (iTr-r24-0293) = (1 * 1 * 5) / ((0+1) * (293)) = 0.0170648

Value (iTr-r24-0429) = (1 * 1 * 5) / ((1+1) * (429)) = 0.0058275

Value (iTr-r24-0707) = (1 * 1 * 5) / ((2+1) * (707)) = 0.0023573

Value (iTr-r24-0817) = (1 * 1 * 5) / ((3+1) * (817)) = 0.0015300

Value (iTr-r24-1107) = (1 * 1 * 5) / ((4+1) * (1107)) = 0.0009033

The value of the duplicated iTrace messages decreases according to the "F" function defined in "value". The accumulated value for all the useful or useless iTrace messages generated by router 24 is 0.028, according to the "Value" measurement.

Based on the definitions of "usefulness" and "value", four different iTrace schemes are evaluated: the original iTrace with or without the distance factor (the hop-count from victim to the router) and the Intention-Driven iTrace, again with or without the distance factor. Please note that only in the case of multiple victims is distance relevant.

Two different performance measurements--"usefulness (quantity)" and "value (quality)" as defined above--are applied. Due to the space limitation, only the case of multiple attacks versus multiple victims is presented (single-attack vs. single-victim is one special case of multiple-attacks vs. multiple-victims). Figure 6.3 shows the "usefulness" of iTrace messages along "one attack path" against victim 125:

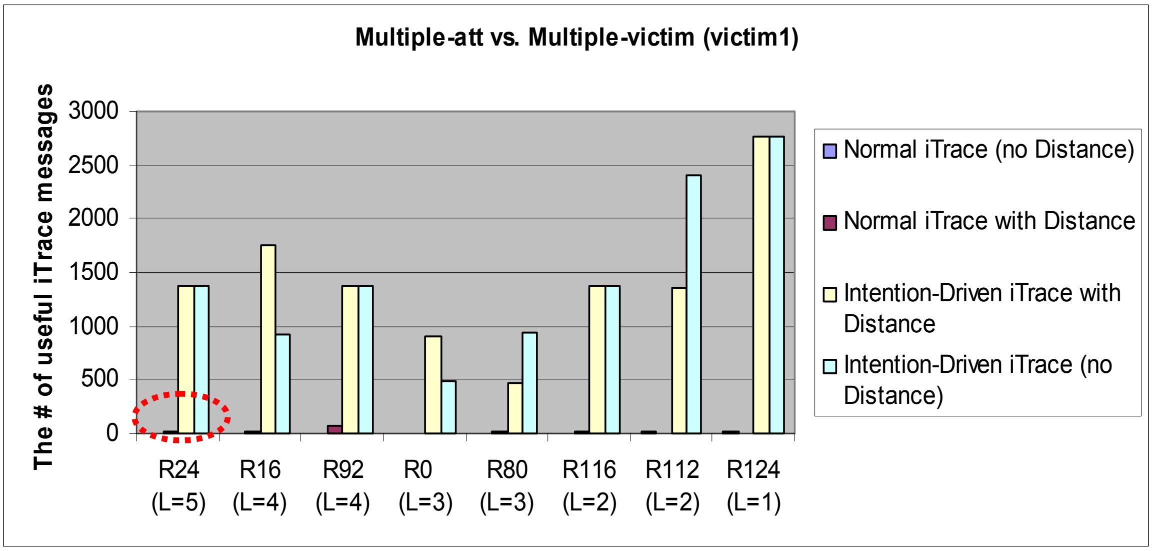


Figure 6. 3 – The comparison of the usefulness of iTrace messages in different iTrace schemes.

Please note that the number of useful iTrace messages generated by the router 24 using the original iTrace scheme is almost to be zero; i.e. even though router 24 generates 1200 iTrace packets, most of those iTrace packets are useless.

In Figure 6.4, we show the same measurements but for a different path against victim 41:

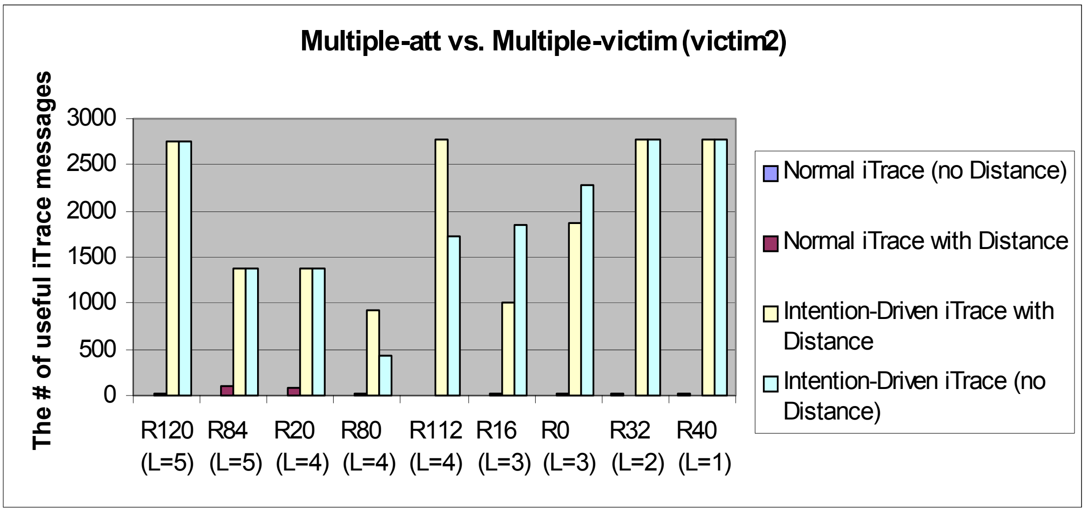


Figure 6. 4 – The comparison of the usefulness of iTrace messages for another path.

These simulation results show that most of the iTrace messages generated by the original iTrace scheme are wasted, while it is not really clear whether the heuristics of "distance" really contributes significantly – under the "usefulness" measurement. However, if we look at the "value" measurements (Figure 6.5), the story is different.

From the results of "accumulated" iTrace message value over time, the conclusions are that: (1) with Intention-Driven iTrace (regardless of distance heuristics), the accumulated value of iTrace information increases very quickly to its maximum in the first few iTrace triggers. This result is very encouraging because it implies that the Intention-Driven iTrace scheme can provide victims with useful iTrace messages very quickly after the attack starts. (2) with the distance heuristics, the value is about 40% better, based on the value measurement (Figure 6.5).

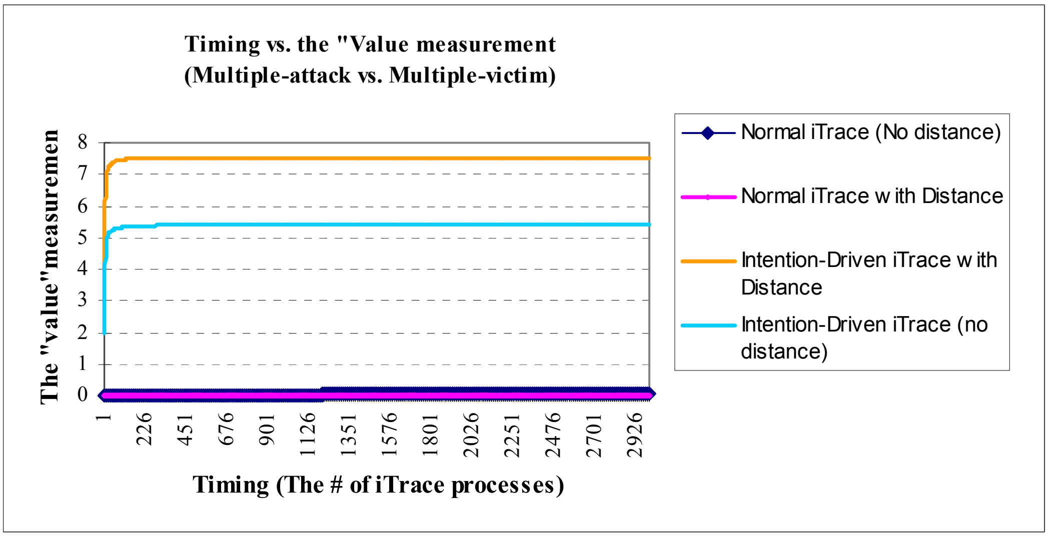


Figure 6. 5 – The comparison of "value" among different schemes on R112.

The simulation results explicitly show that Intention-Driven iTrace scheme has the ability to generate more "useful" and "valuable" iTrace messages toward victim(s) or hosts which really desired to receive iTrace messages. The results also confirm our observation: if the attack traffic rate is relatively low, the original iTrace scheme is almost unable to sample the relatively small attack traffic; and thereby is not able to provide any path information for tracing attacks. The Intention-Driven iTrace scheme (the enhanced version of the original iTrace scheme) has been evaluated in terms of its quantity (usefulness of generating iTrace messages) and its quality (value of quickly generating useful iTrace messages), and the results show that the Intention-Driven iTrace scheme can solve the problem of the original iTrace scheme, even though the attack rate is relatively low.

6.4 Strategy II: The evaluation of Hybrid iTrace scheme

The second strategy focuses on evaluating the Hybrid iTrace scheme. We set up the attack path as follows: 25->24->16->0->112->124->125.

Since router 24 is the one near the attack source node 25, we are more interested in simulating the iTrace generation of router 24. We run the simulation in two ways. On router 24, first we keep the background traffic rate constant (100k packets per second) and vary the attack rate (1k, 5k, 10k, 20k, 50k, 100k, 200k). Second, we keep the attack rate constant (1k packets per second) and vary the background traffic (200, 500, 1k, 10k, 100k, 200k). Initially, we keep the controlled probability ***q*** to be 0.75 and run the simulation. We compare the Hybrid iTrace scheme's capability of generating useful iTrace (quality and quantity) to that of the ID-iTrace and original iTrace schemes. We also evaluate the controlled probability by setting the background traffic rate to be 100k pps and the attack rate to be 500 (the attack rate is relatively low), and then we vary the value of ***q*** (0.1, 0.25, 0.5, 0.75, and 0.99) to evaluate the capability of generating useful iTrace messages in different value of ***q***.

6.4.1 Simulation results and analysis

The simulation results are shown as follows:

□ Constant attack rate (1k pps) vs. variable background traffic rates (1k, 5k, 10k, 50k, 100k, 200k)

The purpose of this simulation is to evaluate the capability of generating useful iTrace messages when the rate of background traffic increases. Figure 6.6 shows that the percentage of the useful iTrace decreases if the background traffic increases. As the rate of background increase up to 50k pps, the original iTrace scheme almost cannot generate any useful iTace message; but the Hybrid iTrace scheme is still able to use 78% iTrace resources to generate useful iTrace messages. Furthermore, the Hybrid iTrace scheme, whether in the Traffic-Separation mode or Probability-Separation mode, performs much better than the original iTrace scheme. It means that the Hybrid iTrace scheme is still able to dedicate about 78% of iTrace resources for tracing attacks even though the rate of background traffic increases up to 200k pps. Another point to note is that even though the Hybrid solution sacrifices some iTrace resources, it enables us to have more control over the iTrace daemon when compared with the ID-iTrace scheme. The results also show that the capability of generating useful iTrace messages is not much different for running in either the Traffic-Separation mode or the Probability-Separation mode.

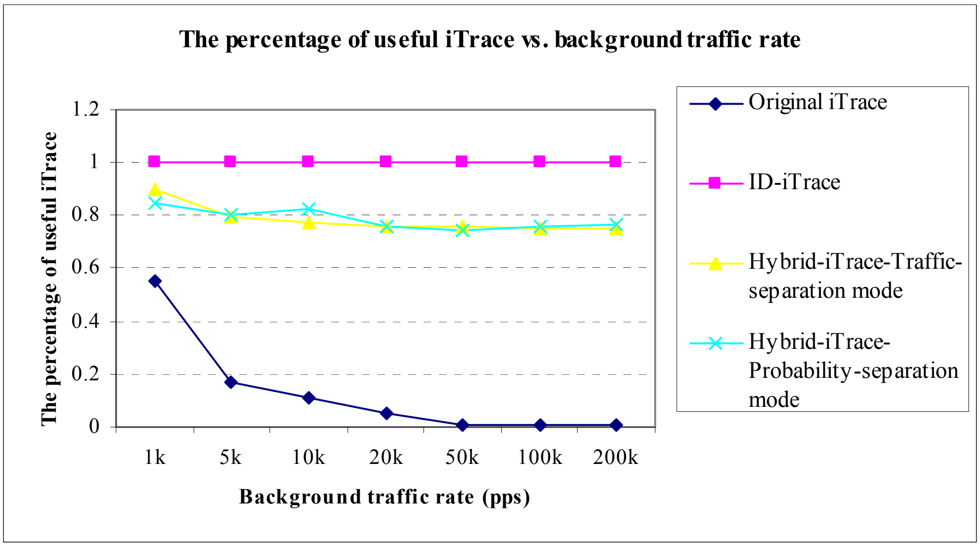


Figure 6. 6 – The percentage of useful iTrace vs. variable background traffic rates.

□ Constant background traffic rate vs. variable attack rates (200, 500, 1k, 10k, 100k, 200k)

Similarly, the goal of this simulation is to observe the behavior of generating useful iTrace messages in different iTrace schemes (the original iTrace scheme, the Intention-Driven iTrace scheme, and the Hybrid iTrace scheme in either the Traffic-Separation mode or the Probability-Separation mode.)

The results (see Figure 6.7) show that while all schemes performed well when the attack rate was high, the Hybrid iTrace scheme performed reasonably well even when the attack rate was very low (200 pps). The simulation results also show us that the capabilities of both the Traffic-Separation mode and the Probability-Separation mode of the Hybrid scheme are almost the same.

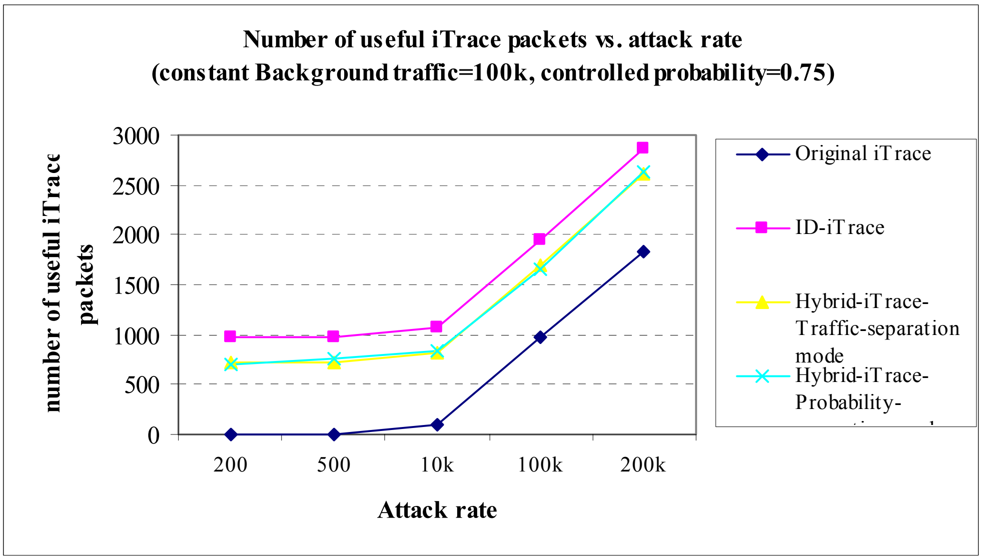


Figure 6. 7 – The number of useful iTrace packets vs. attack rate with constant background traffic=100k, the value of q=0.75.

The previous two results only tell us the quantity measurement (usefulness); however, they cannot provide the view of quality measurement (value). Therefore, we conduct another simulation (background traffic rate = 100k pps, attack rate = 500 pps, and the value of q = 0.75) to collect the accumulated values and analyze the quality measurement of each iTrace scheme.

The results show that the Hybrid iTrace (q = 0.75) scheme is able to generate useful iTrace packets almost immediately. In Figure 6.8, we show the quality measurement of the first 30 iTrace generation; it points out that the Hybrid iTrace and the Intention-Driven iTrace schemes are able to dedicate most iTrace resources to generate useful iTrace messages within the first 30 iTrace generations. Hence the Hybrid iTrace scheme has a very good quality measurement. On the other hand, the accumulated values of the original iTrace scheme are extremely unsatisfactory. That means it generates the first useful iTrace very late (after it generated 240 useless iTrace packets). This shortcoming definitely affects its ability to trace attack sources when the attack rate is low. (As we read the log carefully, the original iTrace scheme only generates three useful iTrace messages: the first one at the 241-th iTrace generation, the second one at 554-th, and the third one at 827-th.)

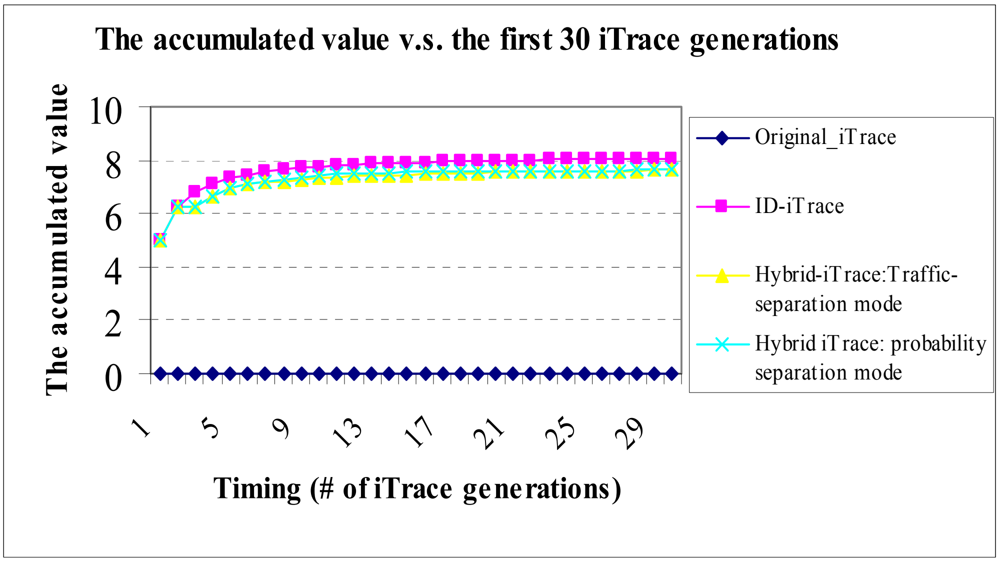


Figure 6. 8 – The accumulation values v.s timing

One of the requirements of the Hybrid iTrace scheme is the ability to control the iTrace daemon. In Figure 6.9, the results show that if we increase the value of ***q***, the capability of generating useful iTrace packets is also increased. These results confirm that the Hybrid scheme, whether in the Traffic-Separation mode or the Probability-Separation mode, is able to satisfy the above requirement. Furthermore, the Traffic-Separation mode performs better if the value of ***q*** is low (***q*** less than 0.25), while the opposite is true of the Probability-Separation mode, though the difference is small. Thus we can select the more effective mode according to the situation.

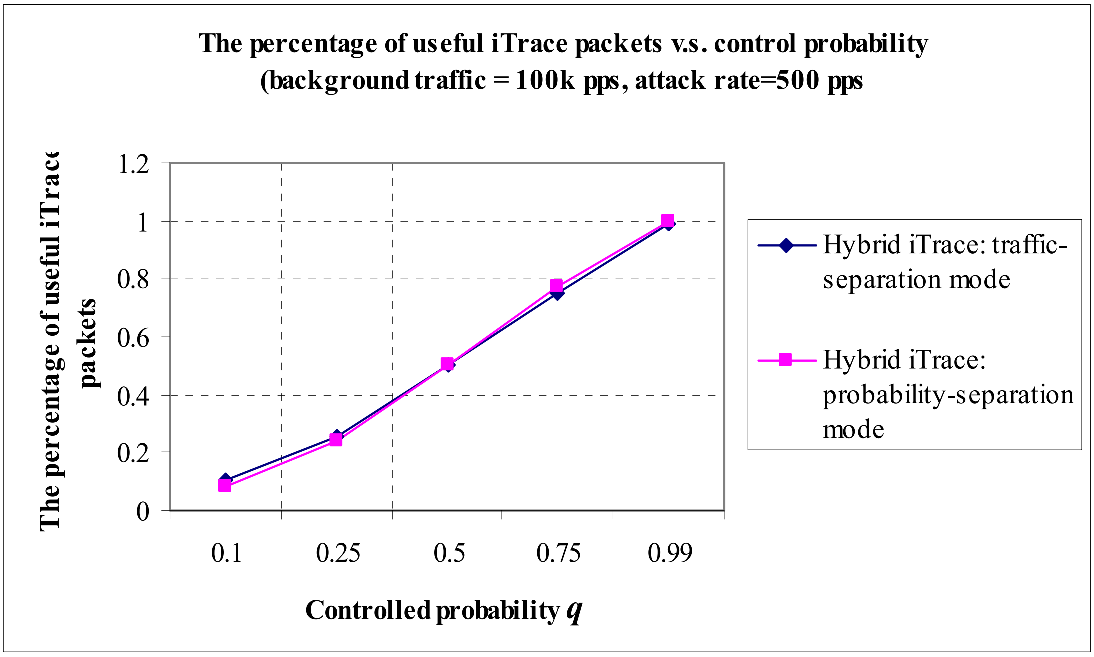


Figure 6. 9 – The percentage of useful iTrace vs. the controlled probability *q*.

6.5 Summary of this chapter

In this chapter, the Intention-Driven iTrace and Hybrid iTrace concepts are simulated in our test-bed. By defining the "usefulness" and "value" of iTrace messages, the routers' capability of generating iTrace messages are measured. The results explicitly show that Intention-Driven iTrace scheme with Intention-bit turned on can greatly improve the capability of the original iTrace proposal in generating useful iTrace messages. In these simulations, the original iTrace scheme's ability to deal with relatively small amount of attack traffic is investigated. The simulation results have proven that the Intention-Driven iTrace scheme (an enhanced version of the original iTrace scheme) is a definite improvement over the original iTrace scheme.

Regarding the concerns of robustness, controllability, and efficiency, we replace the Intention-Driven iTrace design by the Hybrid iTrace scheme: it has almost the same capability of quickly generating useful and valuable iTrace messages and has more control to satisfy different situations.

Two different implementations of Hybrid iTrace scheme (Traffic-Separation mode and Probability-Separation mode) are evaluated using two different methods. First, the ability of the router to generate useful itrace messages is simulated with constant background traffic rate (100k packets per second) and variable attack rates (from 200 pps to 200k pps). The results show that the Hybrid iTrace scheme (in both modes) always has excellent performance in sampling attack packets whether the attack rate is low or high, while the original iTrace scheme gets very low probability of sampling attack traffic when the attack rate is low.

The capabilities of both the Traffic-Separation mode and the Probability-Separation mode are evaluated. Both have almost the same performance, although trivial differences do exist. Of course, if both have the same performance, we would choose the Probability-Separation mode as our implementation reference since it is much easier for us to implement, though ultimately the choice is up to the user, because we recognize that other factors might come into consideration.

In this simulation study, we have addressed the issues of reducing resource wastage and the ability of the iTrace scheme to effectively control iTrace resources. The Hybrid iTrace scheme has been designed to overcome the shortcomings of the original iTrace scheme. Furthermore, it is more robust than the Intention-Driven iTrace scheme. Thus the Hybrid iTrace scheme is the most effective solution to the IP traceback problem.

Chapter 7 Conclusion and Future Work

The IP traceback problem could be solved by either the IPSec-based traceback scheme or the sampling-based traceback scheme. In the IPSec-based approach, the DECIDUOUS tracing model (utilizing IPSec node-authentication technique) was able to trace attack sources within one administration domain. However, tracing across different domains is mandatory in most cases. With the PHIL/PHIL-API and PHIL-switching support, it has been successfully demonstrated that DECIDUOUS is able to trace attack sources across different administration domains.

In the sampling-based approach, the ICMP traceback (iTrace) proposal in IETF was considered since it could potentially trace the reflector-style DDoS attacks and allow more information space to carry attack packet information. After investigating the iTrace scheme, we found that the iTrace scheme wasted many iTrace resources in generating useless iTrace messages. It became worse as the attack traffic decreased. To overcome this problem, the Intention-Driven iTrace (ID-iTrace) was developed to enhance the original iTrace scheme. From the simulation results, we ascertained that the ID-iTrace scheme was able to improve the original iTrace scheme and fix the problem in handling the relatively small attack traffic.

Although ID-iTrace is so powerful, there are major concerns in using ID-iTrace scheme: the robustness, the flexibility and controllability, and the efficiency. Thus, we propose a trade-off solution: Hybrid iTrace scheme. In the normal situation, we can control the iTrace daemon to have minimum capability to generate iTrace messages; on the other hand, we can have higher capability of generating iTrace messages by adjusting the controlled probability ***q***. The simulation results show us that the Hybrid iTrace has very good performance in generating useful and valuable iTrace packets and also provides a feasible control interface for system administrator to flexibly use/control iTrace resources in different situations.

7.1 Contribution Summary

The contributions of this research are listed as follows:

- There existed missing links between applications and IPSec framework. The PHIL and PHIL-APIs provided a socket mechanism for developers to create application-to-application communication, based on the IPSec infrastructure.
- The PHIL-switching technique extended and improved the DECIDUOUS to trace attack sources across the Internet.
- The weaknesses of the original iTrace scheme were investigated and simulated. The simulation results also confirmed our findings.
- To solve the original iTrace problems, the Intention-Driven iTrace was proposed. The simulation results proved that the ID-iTrace scheme is able to improve the original iTrace scheme and fix the problems of handling low attack traffic.
- To provide a robust, efficient and flexible iTrace mechanism, the Hybrid iTrace scheme was proposed; and the simulation results show that the Hybrid iTrace is an effective and feasible solution.
- A light-weight authentication scheme was designed and developed for Intention-Driven iTrace scheme to prevent forging iTrace messages.

7.2 Future work

In this dissertation, the Hybrid iTrace/ID-iTrace schemes are proposed as our core solution for IP traceback problem. Through the probability analyses and simulation studies, the capability of the Hybrid iTrace and Intention-Driven iTrace schemes to trace attacks has been confirmed. For further developments in this direction of research, we plan future work along the following lines: (1) Design and develop an efficient detection algorithm (based on the received iTrace messages) to obtain useful tracing information for IDS to trace attack sources. (2) Design and develop a mechanism to efficiently distribute the Intention-bit information. We may also need a way to pre-distribute the desire of receiving the iTrace messages not only to prevent attacks but also to enable the victim to know immediately where the attackers are in case an attack really happens. (3) Integrate the Intention-Driven iTrace scheme with PHIL-switching scheme. We understand the limitations of Intention-Driven iTrace/Hybrid iTrace schemes in the low throughput network. We also know that the PHIL-switching is able to efficiently identify attacks in local administration domain. The PHIL-switching also provides the design-in functionality of Intrusion Response System (IRS). If the Intention-Driven iTrace/Hybrid iTrace schemes can trace and narrow down the attack sources to a reasonable small domain and pass this information to PHIL-switching supported DECIDUOUS, then different local DECIDUOUS daemons can simultaneously identify the attack sources in detail. With the design-in IRS system of PHIL-switching, the local DECIDUOUS is able to respond immediately by either monitoring the suspect flow or dropping the attack flow to reduce the effects of this attack flow on the victim. To combine and integrate these two schemes might provide an effective solution to tracing DoS/DDoS and reflector-style DDoS attacks.

Bibliography

1. Denial-of-Service (DoS) attack resources, < http://www.denialinfo.com/>
2. Gary C. Kessler, “Defenses Against Distributed Denial of Service Attacks”, SANS Institute, pp. 1-3, Nov. 29, 2000
3. J. Mirkovic, J. Martin, P. Reiher, “A Taxonomy of DdoS attacks and DdoS Defense Mechanisms”, Technical Report #020018, Computer Science Department of UCLA.
4. NIPC (National Infrastructure Protection Center):<http://www.nipc.gov/>
5. Howard. “An analysis of Security Incidents on the Internet 1989-1995.” Ph.D. thesis, Carnegie Mellon University, April 1997.
6. Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson “Network Support for IP Traceback”, IEEE/ACM Transactions, November 2000
7. CERT Coordination Center (CERT/CC) Advisory -- Denial of Service attacks: <http://www.cert.org/tech_tips/denial_of_service.html.>
8. David Moore, Geoffrey M. Voelker, and Stefan Savage, “Inferring Internet Denial-of-Service Activity,” Proceedings of the 2001 USENIX Security Symposium, Aug. 2001.
9. Richard Power. “Computer Security Issues & Trends – 2002 CSI/FBI Computer Crime and Security Survey”, Volume VIII, NO.1, Spring 2002.
10. DDoS Attacks Cripple Yahoo, CNN, Amazon and Buy.com Feb. 9, 2000 HackWatch <http://www.hackwatch.com/~kooltek/dosattacks.html>
11. W. Richard Stevens, “TCP/IP Illustrated, Volume I: The Protocols”, p.111-117, Addison Wesley, 1994.

12. S.M. Bellovin, “Security problems in the TCP/IP protocol suite”, Computer Communication, Rev. Vol. 19 no. 2, pp.32-48, 1989.

13. Start Staniford-Chen L. Todd Heberli, “ Holding Intruders accountable on the Internet”, Technical report of Dept. of Computer Science, UCDAVIS

14. Deokjo Jeon, “Understanding DDOS Attack, Tools and Free Anti-tools with Recommendation”, SANS Institute, April 7, 2001.

15. “Characterizing and Tracing Packet Floods Using Cisco Routers”, CISCO Document ID: 13609, February 2003.

16. IP Security Protocol (ipsec), <http://www.ietf.org/html.charters/ipsec-charter.html>

17. ICMP traceback (itrace) <http://www.ietf.org/html.charters/itrace-charter.html>

18. Wu, S. F. C. L. Wu, Zhang L. Massey D. and Mankin A, “Intention-Driven ICMP traceback”, Internet Draft, IETF, Feb. 2001. (Draft-wu-itrace-intention-02.txt)

19. P. Ferguson, “Network Ingress Filtering: Defeating Denial of Service Attacks, which employ IP Source Address Spoofing “, IETF RFC 2267, 1998.

20. Hal Burch and Bill Cheswick, “Tracing Anonymous packets to their Approximate Source”, 2000 LISA XIV, December 2000.

21. H.Y. Chang, S.F. Wu, et al., "Design and Implementation of a Real-Time Decentralized Source Identification System for Un-trusted IP Packets", appeared in DARPA Information Survivability Conference and Exposition (DISCEX 2000), IEEE Computer Society Press, January, 2000.

22. Dawn Xiaodong Song and Adrian Perrig, “Advanced and Authenticated Marking Schemes for IP Traceback,” in Proceedings of IEEE Infocom, Apr. 2001.

23. Xinyuan Wang, Douglas S. Reeves, S.F. Wu, Jim Yuill, "Sleepy Watermark Tracing: An Active Network-Based Intrusion Response Framework, Technical Report, North Carolina State University, 2001.

24. Robert Stone, "CenterTrack: An IP Overlay Network for Tracking DoS Floods", In Proc. 2000 USENIX Security Symposium, July 2000.

25. Drew Dean, Matt Franklin, and Adam Stubblefield, "An Algebraic Approach to IP Traceback", Technical report.

26. S. Kent, "IP Authentication Header (AH)", IETF draft (draft-ietf-ipsec-rfc2402bis-01.txt), July 2002.

27. S. Kent, "IP Encapsulating Security Payload (ESP)", IETF draft (draft-ietf-ipsec-esp-v3-03.txt), July, 2002.

28. Ravindra Narayan, "Socket API Extensions to Extract Packet Header Information List (PHIL)" Master thesis, May 1999.

29. C. L. Wu, S. F. Wu, and Ravindra Narayan, "PHIL (Packet Header Information List): design, Implementation and Evaluation", IEEE/ICCCN2001, October 2001.

30. Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Stephen T. Kent, and W. Timothy Strayer, "Hash-Based IP Traceback," in Proceedings of SIGCOMM, Aug. 2001.

31. Luis A. Sanchez, Walter C. Milliken, Alex C. Snoeren, Craig Partridge, Christine E. Jones, Fabrice Tchakountio, Stephen T. Kent, and W. Timothy Strayer, "Hardware Support for Hash-Based IP Traceback," BBN Technologies.

32. David Moore, Geoffrey M. Voelker, and Stefan Savage, "Inferring Internet Denial-of-Service Activity," Proceedings of the 2001 USENIX Security Symposium, Aug. 2001.

33. Kihong Park and Heejo Lee "On the Effectiveness of Probabilistic Packet Marking for IP Traceback under Denial of Service Attack", Tech. Rep. CSD-00-013, Department of Computer Sciences, Purdue University, June 2000.

34. Adrian Perrig, Ran Canetti, Dawn Song, and Doug Tygar, "Efficient and secure source authentication for multicast," in Proceedings of NDSS, Feb. 2001.

35. Vern Paxson, "An analysis of using reflectors for DDoS attack" At&T Center for ICSI, 2001.

36. Angelos D. Keromytis, John Ioannidis, and Jonathan M. Smith, "Implementing IPSec," IEEE, August 1997

37. D. Maughan, M. Schertler, M. Schneider, and J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)", RFC2408, November 1998.

38. H. Y. Chang, S. F. Wu, et al., "DECIDUOUS: Decentralized Source Identification for Network-based Intrusions", IFIP/IEEE International Symposium on Integrated Network Management, IEEE Communications Society Press., May 1999.

39. Intrusion Detection Resources: <http://www.cerias.purdue.edu/coast/ids/ids-body.html>

40. F. Jou, F. Gong, C. Sargor, S.F. Wu, and R. Cleaveland. "Architecture Design of a Scalable Intrusion Detection System for the Emerging Network Infrastructure." Technical Report E296, Advanced Network Research, MCNC, April 1997.

41. Case J. D. Fedor, M.S. Schoffstall, M. L., and Davin C, "Simple Network Management (SNMP)," RFC1157, May 1990.

42. Case J. D., McCloghrie K, Rose M. T., and Waldbusser S., "An Introduction to Version 2 of the Internet-Standard Network Management Framework," RFC1441, May 1993.

43. Linux kernel http://www.kernel.org/

44. FreeSwan Web site:<http://www.xs4all.nl/~freeswan/>

45. Postel, J. B., et al, "Transmission Control Protocol", RFC793, IETF, 1981.

46. Postel, J. B., et al, "User Datagram Protocol", RFC768, IETF, 1980.

47. Divert socket: <http://www.anr.mcnc.org/~divert/index.shtml>

48. W. Richard Stevens, "Unix Network Programming", p. 332-334, Prentice Hall, 1990.

49. Steven M. Bellovin, "ICMP Traceback Messages", Internet Draft, IETF, March 2001.

50. Jon Postel. Internet Control Message Protocol. RFC 792, Sep. 1981

51. Mills D.L., "Network Time Protocol Version 3: Specification, Implementation, and Analysis," RFC 1305, March 1992.

52. H. Krawczyk, M. Bellare, and R. Canetti "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

53. S. Laih, L. Harn, and C.C. Chang "Comtemporary Cryptography and its Applications", Unalis Corp. 1998.

54. Kaufman, R Perlman, and M. Speciner, "Network Security – Private Communication in Public World", Prentice Hall, 1995.

55. Public Key Infrastructure in IETF, <http://www.ietf.org/html.charters/pkix-charter.html.>

56. Diffie, W., and Hellman, M. "privacy and Authentication: An Introduction to Cryptography." Proceedings of the IEEE, March 1979.

57. Diffie, W. “The First Ten Years of Public-Key Cryptography.” Proceedings of the IEEE, May 1988.

58. Postel, J. B., et al, “Internet Protocol,” RFC791, IETF, 1981

59. C.L. Wu, X.L. Zhao, S. Felix Wu, Wayne Huang, Dan Massey, Allison Mankin, Lixia Zhang, “On Design and Evaluation of Intention-Driven ICMP traceback”, IEEE/ICCCN2001, Oct. 2001.

60. K. Lougheed, Y. Rekhter, “A Border Gateway Protocol (BGP) “, RFC1105, June 1989.

61. Srihari R. Sangli, Daniel Tappan, Yakov Rekhter, “BGP Extended Communities Attribute ”, draft-ietf-idr-bgp-ext-communities-05.txt, May, 2002

62. Manning, “Registering New BGP Attribute Types “, RFC2042, IETF, January, 1997.

63. Unix Network Programming Volume 1: Networking APIs: Sockets ans XTI, 2nd edition, Chapter 25, Prentice-Hall, 1998.

64. Network Simulator NS-2: <http://www.isi.edu/nsnam/ns/ >

65. Marc Greis’s NS2 tutorial:<http://www.isi.edu/nsnam/ns/tutorial/index.html>

66. TCL/TK developer exchange: <http://www.tcl.tk/software/tcltk/>

67. iTrace module in NS2, <http://bone.csc.ncsu.edu/~cwu4/simulation>