NAIST-IS-DT0361029

Doctoral Dissertation

Studies on Interconnection Architecture for Traceback Systems in Practical Network Operations

Hiroaki Hazeyama

Department of Information System
Graduate School of Information Science
Nara Institute of Science and Technology
March 2, 2006

Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Hiroaki Hazeyama

Thesis Committee: Suguru Yamaguchi, Professor
Hideki Sunahara, Professor
Jun Murai, Professor
Youki Kadobayashi, Associate Professor

Abstract

In order to defend attacks on the Internet, traceback techniques are required as well as attack detection techniques and attack protection techniques. Network operators can manually trace back attacks, but it is likely to be a tedious process requiring much time.

Many researchers have proposed various traceback techniques to automate the manual traceback; however, they have faced several difficulties. The difficulties of a traceback are categorized into the inter-domain issues and the intra-domain issues. On the inter-domain traceback, the issues arise from the difficulties in overcoming the barriers on network operation boundaries, especially the leakage of sensitive information, the violation of the administrative permission, the difference of employed techniques among Autonomous Systems (ASes), and the cooperation among ASes. On the other hand, the challenge of the intra-domain traceback is how to detect the attacker node inside a network domain even when both the source IP address and the source MAC address of an attack packet are spoofed. The interconnection between layer 3 traceback techniques and layer 2 traceback techniques is also an issue of the inter-domain traceback because most traceback techniques on each layer cannot track attacks on other layer.

In this doctoral dissertation, we propose an architecture to interconnect different traceback systems for tracing attacks beyond the operational barriers on the inter-domain network and for tracking attacks narrower down from layer 3 traceback to layer 2 traceback on the intra-domain network.

For the issues of the inter-domain traceback, we propose InterTrack as an autonomous traceback architecture to tie the traceback operation to the other network operation. The inter-domain traceback on InterTrack is composed of the federation of intra-domain traceback trials on each AS. InterTrack achieves the inter-domain traceback by the phased tracking approach and the modularization of traceback techniques.

The phased tracking on InterTrack is composed of four stages: "the Tracking Initiation Stage", "the Border Tracking Stage", "the Inter-AS Tracking Stage" and "the Intra-AS Tracking Stage". The tracking initiation stage authenticates traceback clients such as Intrusion detection systems, and controls the request rates from these clients. The border tracking stage preliminarily investigates an issued attack on the borders of an AS to find upstream neighbor ASes which forwards the issued attack, and to check the necessity of the deep internal inspection. Inter-AS tracking stage is composed of the recursive border tracking stage on each upstream ASes. The result of the recursive border tracking among upstream ASes is delivered to the issuer AS as the reverse AS path of the attack. Constructing the attack path by the reverse AS path, InterTrack can conceal the sensitive information of an AS to other ASes while the victim AS knows the ASes on the attack path in order to contact for filtering on the edge networks. InterTrack provides modular components and APIs to use any kind of traceback systems as modules. Because of this modularization, ASes can apply several traceback techniques on its intra-domain network according to their characteristics.

In parallel with the inter-AS tracking stage, each AS can autonomously run the intra-AS tracking stage in order to detect the attacker nodes inside its network domain. We also propose a layer 2 traceback technique and a binding technique between a layer 3 traceback and our layer 2 traceback as a traceback technique on the intra-AS tracking stage. Our layer 2 traceback technique is an ingress-port-based tracking (IPBT) method, which uses the Bloom filter technique and a Forwarding database of layer 2 switches for tracking an Ethernet frame on a layer 2 network. The IPBT method can track a single packet on a layer 2 network despite any source spoofing technique. L2-SPIE is an extension of a Source Path Isolation Engine (SPIE), which is one of hash-digest-logging methods for a layer 3 traceback. L2-SPIE interconnects the layer 3 traceback of SPIE and the layer 2 traceback of IPBT through two kinds of Bloom filters and two convert tables. L2-SPIE can trace back an attack packet both on layer 3 networks and layer 2 networks even when both the source IP address and the source MAC address of the attack packet is spoofed.

Both InterTrack and L2-SPIE were evaluated with implementations. From the results of experiments and the comparison with the evaluation results of other traceback architectures, we cleared that InterTrack interconnects traceback systems of ASes in a sufficiently short time as well as other traceback architectures. In addition, the bottleneck point on the InterTrack is the border tracking stage. We roughly estimated

the response time of a traceback query in 9hop AS length with hash-digest-logging method. The estimated response time of a traceback query on the original issuer AS was about 16 seconds in 9hop AS length. On the evaluation of L2-SPIE, L2-SPIE interconnects a single packet layer 3 traceback and a single packet layer 2 traceback in more or less 1,500 micro-seconds.

This dissertation's proposals on the interconnection architecture for traceback systems will achieve the infrastructure for the inter-domain traceback in the practical network operations, and will bring the traceability on the Internet.

Keywords:

Distributed Denial of Service attack (DDoS), traceback, inter-domain, intra-domain, interconnection, network security, Internet

Acknowledgments

I cannot hope to enumerate, let alone repay, all those to whom I am indebted for assistance not only in preparing this manuscript, but also in helping me to survive and prosper during my years at NAIST. Despite the unavoidable omissions, I still insist on mentioning a few people in print.

First, I wish to express my gratitude to professor Suguru Yamaguchi, my adviser and Committee Chair, for his support, advice, and encouragement. Without his kind cooperation, I could not have completed this dissertation. I would also like to thank professor Hideki Sunahara for his helpful support on my committee. His useful comments helped me to finish my work. I wish to thank professor Jun Murai of Keio University for his long sustained support of my work. His kind support encouraged me to make my research a fruitful one. My special thanks go to associate professor Youki Kadobayashi for his continuous support, assistance, and his great patience. He read my research papers again and again, and gave me invaluable comments.

I discussed the attack traceback architecture with assistant professor Masafumi Oe of the National Astronomical Observatory of Japan and Mr. Ryo Kaizaki of Keio University, as well as, Dr. Kadobayashi many times. Through the numerous discussions with them, I could write this dissertation.

I also thank assistant professor Takesi Okuda and assistant professor Shigeru Kashihara of Nara Institute of Science and Technology, assistant professor Katsuyoshi Iida of Tokyo Institute of Technology for their congenial assistance. Their continuous interest in this research pushed me to my goals.

I would also like to thank associate professor Motonori Nakamura of Kyoto University and members of his laboratory when I was in Kyoto University as a Bachelor's Degree. They gave me the first opportunity to learn the technologies of the Internet and told me about the joys of research. If I had not met them, I would not have learned the Internet technologies and would not have come to Nara Institute of Science and

Technology.(I thank you would have learned Internet technologies nonetheless)

In developing the prototype implementation of the InterTrack, the fellows of Yokogawa Electric Corporation helped me with the design, the test schedule, and task management. Thanks also to Reisei Fujiwara of the Solution Crew, and Mitsuaki Akiyama and Takanori Kawamoto who helped me refactor the prototype implementation for release. Associate professor Yutaka Nakamura of Kyusyu Institute of Technology, Daisuke Miyamoto, Toshiyuki Kubo, Takuji Iimura, and Mio Suzuki kindly answered my questions on the coding techniques in C languages. Their answers provided me with coding skills.

I gratefully acknowledge Interop Tokyo NOC team members, members of CMP Japan (Medialive Japan), and the contributor specialists and STM members who help me catch up on the current network operation style, the cutting edge of network products, and the trends in commerce and research. The experience on the operation on Interop Tokyo from 2003 to 2005 gave me skills and let me realize the importance of my research area.

I am grateful to IPLAB members and my friends. Talking with them was refreshing me and helped to push me toward my goal. I really appreciate their long-lasting friendships.

Finally, I wish to thank my family who are always there to cheer me up every time I am exhausted and want to give up. Their sincere encouragement as well as their love and support has allowed me to gain my degree.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Denial of Service attacks (DoS attacks) and Distributed Denial of Service attacks (DDoS attacks) are serious problems on the Internet. They attack the target site or the target servers by flooding attack packets, which are often forged the source IP addresses or by a single packet including a malicious code to cause buffer-over-flow. A single packet attack stops the service of the target node. A flooding type DoS attack or a DDoS attack consumes server resources and network resources not only the target site, but also on every AS which is included by the forward path of the attack [1]. A WorldWide ISP Security Report by Arbor Network [2] shows that 65 % of IP network operators who participated the survey, said that DDoS is the most significant operational security issue they face today. According to the Arbor Network's report, all tier 1 providers reported sustained attacks larger than 1 Gbps in the second and third quarters of 2005, with several attacks in the 5 to 8 Gbps range. Most tier 2 and content providers (85%) reported that the largest attacks in the same term ranged from 500 Mbps to 1 Gbps. The majority of all survey respondents reported an average of 10 or more actionable attacks per month that significantly impact customer availability, with an average of 40 general attacks per month.

Filtering is one of the effective countermeasures against DDoS attacks. Ingress Filtering [3, 4] and uRPF (unicast Reverse Path Forwarding) [5] are used to prevent source address spoofing, but these are typically useful only at edge networks. The Spoofer Project encourages ingress filtering by distributing check tools in several platforms [6], however, the project reported 25.7% of all ASes still *spoofable* [7]. Although filtering near the victim site can stop the attack traffic, making an appropriate filter is

difficult because of the source spoofed characteristic of a DDoS attack. Filtering near the victim site may sacrifice other customers' service traffic if service traffic matches the filter rule; therefore, filters must be applied in the nearest ASes to the attacker nodes.

In order to apply filters on the attacker side networks, a network operator has to track an attack to the attacker side networks. Network operators can manually track an attack, but it is a tedious process because much time is spent both on the inside investigation and on contacting other ASes to request the traceback.

In order to automate or expedite the traceback, many traceback techniques have been studied and proposed. Traceback techniques are techniques that reconstruct the attack path, and locate the attacker nodes by correcting the attack traffic, routing information, marked packets, or audit log of the attack traffic [8]. By reconstructing the attack path even when the attack employs the source IP address spoofing, traceback techniques can detect the direction of the source of attacks, or the location of the attack nodes. Traceback techniques can be categorized into the inter-domain traceback and the intra-domain traceback according to the main goals of each technique.

In the Arbor Network's report [2], it was remarked that traceback and attack mitigation mechanisms need to be deployed ubiquitously across the Internet. Unfortunately, one cannot say that traceback techniques have already been deployed ubiquitously across the Internet, although several practical tracking techniques or traceback products have been available. There are technical issues on traceback techniques which should be solved.

1.1 Issues on Traceback Techniques

The technical issues of traceback are categorized in the inter-domain issues and the intra-domain issues. The inter-domain issues are mainly caused by the barriers on network boundaries and the routing operation manner. On the other hand, the issues of intra-domain traceback techniques arise from the strict assumptions on the behaviors of attackers and of attacker nodes.

1.1.1 Issues of the Inter-domain Traceback

Unfortunately, no traceback techniques, which can reconstruct the attack path across several network domains, are employed or practiced in the real network operation yet. The difficulties of deploying inter-domain traceback techniques are derived from such operational issues as follows:

- *The risk of exchanging sensitive information about the inside of each network domain.*

  Leakage of detailed backbone topology is a serious problem on a network operation. Especially, if IP addresses of core routers or traceback agents on an AS are revealed to attackers, attackers will hijack or attack the AS's backbone network.

- *The fear of misuses of the traceback technique on each network domain by others.*

  In addition to the leakage of sensitive information, misuses of traceback systems waste resources on each AS. In the worst case, a misuse of traceback systems brings an Internet-wide DDoS attack. Furthermore, the traceback operation has been closely tied to ISP backbone network security. Arbitrary trials of traceback by unauthorized people would not be acceptable to most ISPs.

- *The risk of depending on an unique traceback technique.*

  Each traceback technique has pros and cons [8]. Even if a specific inter-domain traceback technique is well deployed, attackers will develop evasion attacks sooner or later. Also, in order to run a specific inter-domain traceback technique, several ASes should deploy it at the same time. If other ASes deploy another traceback technique, operators have to contact other ASes after all.

  In practice, many ISPs employ multiple detection and traceback tools or targeted/phased deployments for a given tool set in their networks [2]. From the view of the current traceback operation, depending on a specific traceback technique is not practical.

These operational issues arise when a trial of traceback attempts to expand beyond the network boundaries. Many proposed traceback techniques lack or ignore the boundaries of network operation and the difference of the operational policies among different network domains.

1.1.2 Issues on the Intra-domain Traceback

The goals of the intra-domain traceback are different between the victim-side networks and the attacker-side networks. One of main goals of the victim side is detecting the ingress point border routers which forward attack packets. Finding upstream neighbor ASes is also the goal of the inter-domain traceback, in order to contact neighbor ASes for further trials of traceback or for actions of the attack mitigation. On the other hand, the main goals of the attacker side ASes is to discover the location of attacker nodes. The logical location of attacker nodes is required to isolate attacker nodes from their networks. Network operators may want the physical/geographical location of the attacker nodes for removing malicious codes on the attacker nodes. Several intra-domain traceback techniques are available as free softwares, manufacture products or operation techniques on one network domain. These traceback techniques are categorized into the layer 3 traceback techniques [9–17], and the layer 2 traceback techniques [18–23]. However, each of techniques has a limitation according to the technique and the target.

Regardless of the main goals of each specific technique, the major issues on the intra-domain traceback are :

- How to detect the direction or the location of attacker nodes when the attacker node uses source-spoofed packets.
- How to track back the attack which is composed of a few packets, even when the attack is a single packet attack.
- How long after does a traceback system keep the evidence of the attacker nodes.
- How to interconnect layer 3 traceback techniques and layer 2 traceback techniques.

The layer 3 traceback can locate the nearest edge router to an attacker node, however, most of layer 3 traceback techniques cannot identify upstream neighbor routers or neighbor ASes, or cannot find the location of the attacker node on the layer 2 subnet [24–36].

For detecting the location of a node on a layer 2 network, several management tools can locate a node by checking a MAC address entry on the Forwarding Data Base (FDB) of each layer 2 switch on the layer 2 network [18–23, 37, 38]. This kind of backtracking by MAC address, called the FDB-checking method, can locate a node on

a layer 2 network. In other words, the FDB-checking method can detect the learning interface on each layer 2 switch or Access Point (AP) of the Wireless LAN. The FDB-checking can also reveal the geographical location of the issued node using a database of the layer 2 location with the geographical location of each layer 2 switch or AP such as the management information base (MIB-II) [39]. Even when the source MAC address of an attack packet was spoofed, the FDB-checking method can locate the node because of the characteristics of learning the MAC address on the layer 2 switch. However, the FDB-checking method cannot trace a MAC address when layer 2 switches have already aged out the issued MAC address.

The issue of interconnecting the layer 3 traceback techniques and the layer 2 traceback techniques arises from the fact that the target packet of the layer 3 traceback is usually a source-spoofed packet. Generally, when an attacker node sends a source address spoofed packet without ARP resolution, the gateway router cannot get the MAC address of the attacker node by the ARP resolution because the issued IP address is different from the IP address assigned an interface of the attacker node. Even when the gateway router resolves the MAC address of a source IP spoofed packet, the MAC address might be assigned another node on the same layer 2 network. Moreover, if the attacker node hijacked the IP address and the MAC address of another node on the same layer 2 network and sent a single packet attack with these IP and MAC address, the attacker node can hide its location and put the blame on another node.

Most layer 2 traceback techniques can be available in only one layer 2 network. Thus, a layer 3 traceback technique has to mediate the layer 2 traceback techniques on different layer 2 networks. Unfortunately, the required information to track an attack is usually different on the layer 3 traceback techniques and on the layer 2 traceback techniques. In order to cooperate the layer 3 traceback and the layer 2 traceback, some logging technique is required to bind the source IP address and the source MAC address of an Ethernet frame on each layer 2 network. Of course, logging the source IP address and the source MAC address on each Ethernet frame with the packet signature could track back a single packet attack. However, such logging technique might consume large amount of storage to construct the database. Also, recording the changes of FDB on each layer 2 switch in order to avoid the aging out of FDB consumes much more storage.

The challenge of the intra-domain traceback is how to construct a logging method to interconnect the layer 3 traceback and the layer 2 traceback with reducing the

required storage size.

1.2 Contributions

This dissertation explores a traceback architecture and techniques to interconnect different traceback systems for overcoming issues both on the inter-domain traceback operation and on the detection of an attacker node's location in a spoofed single packet attack on an intra-domain network.

In order to operate the inter-domain traceback in practice, we propose *InterTrack* as an interconnection architecture for traceback systems to overcome the issues on the inter-domain traceback. The one of key ideas of InterTrack is the federation of the intra-domain traceback on each network domain. The inter-domain traceback on InterTrack is composed of the federation of intra-domain traceback trials. By separating the operation realm of an inter-domain traceback according to the network boundaries on routing operation, InterTrack allows each AS to operate traceback trials on its network domain by itself only.

Another key idea of InterTrack is the phased tracking, that is, a traceback trial on InterTrack is phased in four stages; *tracking initiation stage*, *border tracking stage*, *inter-AS tracking stage* and *intra-AS tracking stage*. Each tracking stage reflects operational boundaries on the routing architecture or the network operation.

Tracking initiation stage represents the relation between end users and operators or the operational boundary between edge networks and the backbone network in an AS. Border tracking stage and Inter-AS tracking stage shows the relation among ASes on EGP routing or the the network / routing architecture on EGP domain. Intra-AS tracking stage is in accordance with the operational boundaries on an AS, that is, the relation between the backbone network and the IGP sub-domains on an AS.

The border tacking and the inter-AS tracking achieves a preliminary investigation across ASes for finding the attacker AS, and the intra-AS tracking provides a detailed internal inspection on each AS for locating attacker nodes. The phased tracking of InterTrack can run an inter-AS traceback and intra-AS traceback on each AS in parallel. On the inter-AS traceback, each AS aggregates the result of the border tracking stage in AS hop level. Each AS recursively forwards and replies the result of border tracking stage, the inter-AS tracking makes a reverse AS path of the issued attack which conceals any sensitive information of each AS.

Due to the federation architecture and the phased tracking stages, each AS can loosely cooperate with each other through only messages for the inter-AS tracking stage. This loose cooperation among ASes allows each AS to employ different traceback techniques on the border tracking stage and on the intra-AS tracking stage regardless of traceback systems on other ASes. InterTrack also provides several API and module components to use any kinds of tracking systems and to achieve the tight cooperation among traceback systems, detections systems and protection systems.

We also propose an *ingress-port-based tracking* (IPBT) method as a single packet traceback technique on a layer 2 network. In addition to IPBT method, we present a layer 2 extension approach to a hash-digest-logging technique of layer 3 traceback [29] (L2-SPIE) in order to overcome the issues on the intra-domain traceback. The key idea of our IPBT method is recording the ingress (received) port on each layer 2 switch about each Ethernet frame. By recording the ingress port, IPBT method can detect the location of an attacker node even when the attacker node sent an attack packet which was spoofed both the source IP address and the source MAC address.

As a binding technique between a layer 3 traceback technique and a layer 2 traceback technique, L2-SPIE prepares two tables, one is NI-ID table, the other is Port-MAC table. By these two tables and the Bloom filter method [40], L2-SPIE interconnects hash-digest-logging method on a layer 3 traceback technique and the IPBT method as a layer 2 traceback technique. Using Bloom filter techniques, L2-SPIE reduces the required storage to record the packet information for all Ethernet frames.

Along the way, our proposals make the following specific contributions:

- The loose cooperation among ASes on the deployment traceback systems and on traceback operation that is based on the federated traceback architecture of InterTrack and on the independence of InterTrack from a specific traceback technique.

  The InterTrack architecture were designed for loosening the cooperation among ASes on the deployment of traceback systems. Dividing the operational boundaries of a traceback trial along with the network boundaries on the routing operation allows each AS to operate a traceback trial according to AS's operational policy. Through the module components and the phased tracking apporach of InterTrack, ASes can choose or renew traceback systems for internal use in accordance with the pros and cons of each traceback system or the investment cost,

regardless of the traceback systems on other ASes.

And InterTrack messages were designed to conceal the sensitive information of ASes. The ITM trace reply message contains only the reverse AS path information, and it does not contain the detailed topologies of ASes; therefore, the leakage of sensitive information of each AS will not occured by replying a traceback trial.

- The same manner of the traceback operation as the manner of other network operations by the phased tracking approach of InterTrack.

  The phased tracking approach was designed along with the network operation boundaries on the routing operation. By phasing an inter-domain traceback into four stages, ASes can authorize the users on each stage. ASes can also delegate the internal traceback operation to the organizations on the IGP sub-domains.

- The model of the border tracking technique for identifying the upstream neighbor ASes.

  Based on the assumption about the BGP-peering among ASes, we analyzed the variations of the AS status against a DDoS attack, and we modeled the border tracking technique to aggregate the reverse path of an attack from the router-hop level to the AS-hop level. This model of the border tracking technique is independent from a specific traceback techniques. As a proof-of-concept program, we developed a sample BTM for a hash-digest-logging-based traceback product according to this model.

- The automated procedures on the inter-domain traceback through InterTrack.

  The procedures of a traceback trial on InterTrack was designed to automate the manual traceback over the network operation boundaries. The preliminary experiments of InterTrack showed that the average time of a trial on InterTrack was estimated as 16 seconds with a hash-digest-logging-based border tracking system in 9 AS hop length. Comparison with other inter-domain traceback techniques indicated that InterTrack will automate the manual traceback and shorten the time spent for a traceback trial as well as other traceback techniques do.

- The detailed internal inspection about the source of an attack packet that is produced by the characteristics of the IPBT method.

Our IPBT method can track back a single packet on a layer 2 network to the port to which an attacker node connects despite the source-spoofed characteristic of the IP address and the MAC address.

- An interconnection between a hash-digest-logging method on the layer 3 traceback and the IPBT method on the layer 2 traceback that is provided by L2-SPIE.

  We developed a prototype implementation of L2-SPIE. The evaluation result shows that the average time of an traceback trial on L2-SPIE was about 1,500 micro-seconds.

- The foundation of the self-defending network architecture that would be provided by the cooperation among traceback systems, detection systems and protection systems through InterTrack and L2-SPIE.

  InterTrack and L2-SPIE provide a multi-layer traceback to locate the attacker nodes from the layer 3 traceback to the layer 2 traceback. We also designed InterTrack to cooperate with other detection systems and protection systems for the multi-layer traceback and the attack mitigations.

InterTrack will realize the inter-domain traceback in the practical network operations and automate the communication among any countermeasures of DDoS attacks over the network operation boundaries in a practical way. InterTrack and L2-SPIE will also achieve the foundation of the traceability on the Internet.

1.3 Organization

The remainder of this dissertation presents the details of each proposal. Chapter 2 motivates this work by describing related work in the fields of the intra-domain traceback techniques, the inter-domain traceback techniques and the discussion for the standardization in the Internet Engineering Task Force (IETF). The details of InterTrack in Chapter 3, including the federation model of the inter-domain traceback, the procedure of the inter-domain traceback, the model of border tracking, details of a prototype implementation and the result of preliminary evaluation are described. The IPBT and L2-SPIE are presented in Chapter 4. Chapter 4 also shows the result of a preliminary experiment about the prototype implementation in *L2-SPIE*. Finally, Chapter 5

presents a summary of the dissertation and concludes with a discussion of both specific contributions and general principles that can be extracted from the work.

Chapter 2

Related Work

2.1 Traceback Techniques

Many traceback techniques have been well studied and proposed in the last 6 years. The types of traceback techniques can be divided into two categories: intra-domain traceback and inter-domain traceback.

The intra-domain traceback techniques detect upstream neighbor ASes and downstream neighbor ASes on the attack path, or locate a node, perhaps an attacker node inside a network domain. The intra-domain traceback techniques are categorized in the *Link Testing method*, the *Specialized Routing method*, the *FDB-checking method*.

On the other hand, many traceback techniques have been proposed to archive an automatic inter-domain traceback. They can be categorized in the *Traceback Packet method*, the *Packet-Marking method*, the *Pushback method* and the *Hash-digest-logging method*. These traceback techniques try to reconstruct the attack path across the Internet by collecting the routers' or paths' information from the victim to the attacker nodes.

In this chapter we try to discuss pros and cons of each type of traceback technique.

2.1.1 Intra-domain Traceback Techniques

The intra-domain traceback techniques are categorized in the *Link Testing method*, the *Specialized Routing method*, the *FDB-Checking method*. The intra-domain traceback techniques detect the ingress-point routers, the upstream neighbor ASes and the

downstream neighbor ASes on the attack path, or detect the node inside a network domain.

Link Testing Method

The link testing method watches the change of the volume of traffic flow by applying filters or inputting other traffic. Burch and Cheswick pioneered this method by proposing a novel technique that systematically floods candidate network links in [26]. This method is simple and is processed manually; however, this method easily affects other service traffic because of the packet drop by filtering or the consumption of bandwidth by the inputted test traffic.

Specialized Routing Method

The specialized routing method changes routing tables on border routers to correct all traffic to the victim's site in the centralized watching point or to divide different paths. In these ways, specialized routing methods track the ingress-point routers and the upstream neighbor ASes. In this method, several routing techniques are employed such as IP tunneling [25], black hole routing [9–11], MPLS [41, 42], redirection [43, 44],VPN [32], etc.. Specialized routing methods require considerable knowledge of routing protocols and the internal network topology, therefore, they can be used in only one network domain. Each of them has been available in routers of several vendors, however, and several specialized routing methods have been practiced in several ISP's backbone network. The specialized routing method is useful to the ingress point border routers, upstream neighbor routers on neighbor ASes, however, the specialized routing method cannot identify the attacker node itself.

FDB-Checking Method

The FDB-checking method tracks a node inside a network domain by checking the MAC address entry on the Forwarding Data Base (FDB) of the switches in a hop-by-hop fashion. The FDB-checking method is now available in several manageable layer 2 switches and their management tools or special commands [18–21]. In addition to the single-vendor environment, SNMP-based FDB-checking software or appliances are available for a multi-vendor environment [37, 38] only when the switches on a network domain support the *dot1dTpFdbTable* of the Bridge-MIB [45] or vendor extension MIB

corresponding with the Bridge-MIB. The FDB-checking also can be used in wireless networks with some MAC address filtering method [22,23]. The FDB-checking method can trace spoofed MAC address while switches learn the spoofed MAC address, however, it is difficult to bind a MAC address and the IP address which is used in a source spoofed packet without a binding method based on the log files of an IDS or PCAP [46].

Summary of Intra-domain Traceback Techniques

These intra-domain traceback techniques require dynamic configuration changes of facilities or an administrative permission to access databases; therefore, applying these traceback techniques to the inter-domain traceback is difficult.

If an AS tries to locate an attacker node in the AS's network by the attack packet which was captured in the victim site, some binding method is required to interconnect layer 3 traceback techniques and the layer 2 traceback techniques because the attack packet in the victim site does not have the Ethernet information on the AS's layer 2 network.

2.1.2 Inter-domain Traceback Techniques

To archive the inter-domain traceback automatically, many traceback techniques have been proposed. They can be categorized in the *Traceback-packet method*, the *Packet-marking method*, the *Pushback method* and the *Hash-digest-logging method*. These traceback techniques try to reconstruct the attack path from the victim to the attacker nodes across the Internet. Inter-domain traceback techniques usually reconstruct the attack path in the hop-by-hop fashion, therefore, most of techniques can be employed as the intra-domain traceback technique.

Traceback-Packet Method

Pioneered by Belovin et al. [27], the traceback-packet method samples a packet as a target, generates a traceback packet against the targeted packet and sends the same destination address of the targeted packet. On a DDoS attack, the victim site will receive many traceback packets according to the volume of the attack traffic, therefore, the victim site can reconstruct the attack path by the link information or by information from the router's identifiers written in the traceback packets. A traceback packet

contains link information, router ID, AS information, the targeted packet information, etc.. Several formats of traceback packets have been proposed, for example, ICMP traceback messages [27, 47, 48], NetFlow [12], or sFlow [13]. In order to reduce additional traffic caused by traceback packets, a traceback-packet method usually employs a sampling method. Sawai et al. evaluated the sampling rate, the volume of additional traffic, and the expectation time of attack path reconstruction in several confidential degrees [49].

Despite these contributions, it is difficult to use traceback-packet methods as the inter-domain traceback because traceback packets generated by an AS might become unwanted traffic for other ASes.

Packet-Marking Method

To avoid generating additional packets, several packet-marking methods have been studied [28,50–52]. Most packet-marking methods over-write the router ID or the path information in the IP identification field of IPv4 packets. Packet-marking methods do not generate any additional traffic, but they sacrifice several services which depend on the information of the IP identification field. According to CAIDA Project report of 2001 [53], several tunneling protocols such as GRE, L2TP, IPSec, PPPoE [54–57] or multi-media traffic such as Real Audio or Windows Media easily cause fragmentation of a packet; therefore, such traffic depends on the IP identification field to serialize the sequence of packets. Packet-marking methods sacrifice such service traffic, therefore, the over-writing technique of the packet-marking method is infeasible. Furthermore, in the IPv6 environment, the flow label field, which corresponds to the IP identification field of IPv4, is not an invariant field [58].

AS-Hop Aggregation

Both the traceback-packet method and the packet-marking method may cause a leakage of detailed topologies on ASes. Addressing the leakage of detailed topology, AS-hop aggregation methods have been proposed on both the traceback-packet method [33] and the packet-marking method [59].

In spite of such topology aggregation to avoid the leakage of sensitive information, each method requires a tight cooperation among ASes to collect traceback packets or marked packets, for reconstructing and verifying attack paths and for deploying a

specific technique at the same time.

Pushback Method

Pushback [30] represents a cooperation among firewalls by propagating filter rules in the hop-by-hop fashion. This method can not only trace the direction of the attack traffics but can also protect the attack by filtering. The propagating message was also proposed in IETF [60], however, using pushback in the inter-domain traceback is difficult because the pushback approach forces other ASes to change filter rules.

Hash-Digest-Logging Method

The characteristics of packet sampling or traffic analysis, traceback-packet methods, packet-marking methods or pushback methods are difficult to use to trace the attack path of a single-packet attack. On the other hand, the hash-digest-logging method has the capability to trace single-packet attacks. The hash-digest-logging method records all packets traversed on each capture point with reducing required storage size by hash digesting or Bloom filter techniques [29, 31]. On a traceback trial, the attack path is constructed by checking the records on capture points. Several products of the hash-digest-logging method are available [15–17]. To reduce the false positive rate, Shanmugasundaram et al. have proposed a hierarchical Bloom filter [61]. In addition to this, several researchers have already studied Bloom filter usages on the high speed link over 10Gbps such a Bloom-filter-based exact matching algorithms [62], or a hybrid method of sampling and hash-digest-logging [63].

As well as other inter-domain traceback techniques, the hash-digest-logging method has a difficulty on deployment. Although Gong et al. [64] discussed the AS-level partial deployment scenario of Source Path Isolation Engine (SPIE) [15], which is one of hash-digest-logging methods, the inter-domain traceback highly depends on SPIE and its protocol. The hash-digest-logging method requires the initial investment for deploying capture-point appliances and consumes splitters or mirror ports on layer 2 switches to input all packets into each capture point. In addition to the costly investment, each of hash-digest-logging methods does not have compatibility, replaceability or affinity with others. The dependence on only a hash-digest-logging method cannot deal with evasion attacks. The deployment of the hash-digest-logging methods is difficult because of the lack of replaceability or affinity with others or the fear of evasion attacks.

Summary of Inter-domain Traceback Techniques

In sum, no inter-domain traceback techniques have been deployed or practiced in a real network operation because of the following reasons; the possibility of leakage of topology, the lack of replaceability or affinity with another traceback technique, or the operational cost or budget for deploying an IP traceback architecture with several ASes together.

2.2 Discussion in IETF for the Practical Use of Inter-domain Traceback Techniques

Several IP traceback architectures, standard message formats, or protocols have been discussed in the Internet Engineering Task Force (IETF). Unfortunately, these proposals are still discussed in Internet-Drafts or have already expired [27, 47, 48, 65–67]. ICMP Traceback Messages and their extensions were proposed in the iTrace working group [27, 47, 48]. Partridge et al. proposed a traceback message protocol based on their Source Path Isolation (SPIE) Architecture in 54th IETF meeting and discussed the issues on IP traceback in the Internet at the IPPT BoF at the same meeting [65]. Also, at the same IETF meeting, Keeni et al. proposed their IP traceback architecture based on hash digests, which is different from SPIE in the query method [66]. Oe et al. also presented a hierarchical IP traceback architecture based on a sampling method to track DDoS attacks in the AS-level at the 54th IETF meeting [68]. Each of these IP traceback architectures depends highly on some specific traceback technique; therefore, these architecture cannot be replaced to a new and feasible traceback technique.

Moriatory has proposed the RID (Real-time Inter-network Defense) as an extension of the IODEF message for the interconnection among traceback systems on the inter-domain traceback trials, for the request of filtering or other actions to other ASes, and for reporting the result of a traceback as an incident report to CERT. The architecture and standard message format of RID have been discussed in the INCH working group [67]. As an extension of the IODEF [69], RID architecture does not depend on a specific traceback technique. Kai et al. presented an implementation of RID with several extensions for the traceback on the Japanese local government network in 59th IETF meeting [70] and its preliminary evaluation result in 61th IETF meeting [71]. Due to an extension of the IODEF message, an RID message should be submitted to

CERT as an incident report. Because a recent DDoS attack is composed of several malicious traffic pattern, the victim site operators will make many RID messages for tracking a DDoS attack. Therefore, a traceback by RID may flood the CERT IODEF report archive with numerous RID messages attached the same ID.

2.3 Summary

This chapter discussed pros and cons of each type of traceback technique, and analyzed the issues on traceback techniques. The issues of traceback techniques are summarized as follows:

- Many intra-domain traceback techniques have been available in a practical operation.
- In order to locate attacker nodes on the attacker side network by the attack packet captured in victim's site, an interconnection mechanism between traceback techniques on different layers.
- Many inter-domain traceback techniques have not been available yet since these proposals have the possibility of leakage of topology, the lack of replaceability or affinity with another technique, or the operational cost or budget for the deployment.
- Many inter-domain traceback techniques can be used as intra-domain traceback techniques.
- Although several proposals try to interconnect traceback systems along with the network boundaries, these proposals have difficulties in the practical use.

Considering these facts and issues, an interconnection architecture or an interconnection mechanism between traceback systems are required to solve issues both on the intra-domain traceback and on the inter-domain traceback. The challenges on achieving an interconnection architecture for traceback systems are as follows:

- Designing the architecture to reflect the network operational boundaries.
- Designing the architecture to be independent from a specific traceback techniques.

- Designing the glue of different traceback systems to trace packets in different layers or in different network domain.
- Automating procedures of exchanging traceback information among network domains while concealing sensitive information among network domains and preventing the violation of the administrative permission.

This dissertation challenges these topics to achieve a practical interconnection architecture for traceback systems.

Chapter 3

An Interconnection Architecture for the Inter-Domain Traceback

In this chapter, we present InterTrack, an autonomous architecture in order to make the inter-domain traceback practically usable. The key ideas of InterTrack are as follows:

- a hierarchical architecture according to the network operation boundaries
- phased-tracking and the federation of internal traceback trials for the inter-domain traceback
- concealment of sensitive information on exchanging traceback information among ASes
- modular components and APIs for independence from a specific technique, for multi-layer tracebacks and for self-defending mechanisms

This chapter is organized by sections. First, several assumptions on traceback are given in Section 3.1. The goals of InterTrack are discussed in Section 3.2. We also define the requirements of an inter-domain traceback architecture in Section 3.3 along with assumptions and goals. Following the overview of the architecture of InterTrack in section 3.4, we elaborate the AS path reconstruction mechanism of InterTrack in section 3.5. In section 3.6, the feasibility of InterTrack is discussed. In section 3.7, we describe the details of the prototype implementations of InterTrack. Next, in section 3.8, we evaluate the architecture and the prototype implementation. We also

compare InterTrack with other inter-domain traceback architectures in section 3.9. Finally, we summarize this chapter in section 3.10.

3.1 Assumptions

At the beginning of the discussion about InterTrack, we assume several assumptions about ASes on the traceback:

- Each AS has multiple tools of detection, of traceback, or of prevention for its own network domain.
- Each AS does not want to allow other people to operate or investigate its own network without permission.
- Each AS cannot investigate the inside of its customer's network without permission by the customer.
- Each AS does not want to reveal inside information to others without some reasonable procedures.
- Each AS does not want to be bothered by the traceback queries when the AS is not included in the attack path.

The first assumption is in accordance with the arbor network's security report [2] and with the fact that many implementations or techniques for the inter-domain traceback have already been made available [9–23]. The next two assumptions speak to the administrative permission to operate a network domain. The fourth assumption reflects the confidential or sensitive information on a network operation. The internal information of a network domain is likely to be secret, and if other people want to get such secrets, they would have to sign a non-disclosure agreement or file in a court procedure. The last assumption is used here because a traceback trial puts a high cost on human resources, network resources, or server resources; therefore, broadcasting a request message or additional traffic for a traceback to other ASes regardless of their status against the issued attack is undesirable.

3.2 The Goals of InterTrack

On designing InterTrack as a practical interconnection architecture for traceback systems or as an inter-domain traceback architecture, we accomplish several goals. These goals include:

- Detecting the upstream neighbor ASes of the attack.
- Reconstructing the reverse AS path of the attack.
- Automating the procedures to request a traceback to other network domains.
- Interconnecting any countermeasures of DDoS attacks to expedite attack protection.
- Protecting or isolating attacker nodes on an attacker-side AS with the AS own decision and operation policy.
- Achieving these five goals along with the five assumptions mentioned in section 3.1.

The first three goals are for the traceback trials on InterTrack. InterTrack employs various traceback techniques to investigate inside an AS; therefore, InterTrack must be effective in tracking attacks as a manager component for controlling and working various traceback techniques together in only one AS. Considering the leakage of sensitive information, InterTrack must reconstruct the reverse AS path instead of the actual attack path expressed by router hops. The effectiveness of aggregation router hops to AS hops have already been mentioned in [33] and [59]. InterTrack must also automate the procedures to request a traceback to other network domains. The manual operation on the procedures required to ask a traceback to other network domain spend a lot of time and manual traceback is likely to be finished before the attack finishes or the attack pattern changes. If InterTrack automates these procedures, the time spent for a traceback will be shorter.

The fourth goal of InterTrack is to cooperate with various countermeasures of DDoS attacks. Recently, several attack mitigation products work together in a single vendor environment [72], or among several vendors through a specific data format or API [73]. In order to achieve the correlation among multi-vendors' attack mitigation products,

each vendor has to develop interfaces or a new protocol for other vendor products. An interface and several messages or protocols are provided by InterTrack for reducing such developing overhead.

The last two goals come from the assumptions about AS's network operation policies. In Chapters 1 and 2, we analyzed that the ignorance of network operation boundaries is the major reason why various inter-domain traceback techniques have not been practiced yet, despite the number of proposed inter-domain traceback techniques. InterTrack must be designed along with the assumptions described in Section 3.1.

3.3 Requirements

According to assumptions and goals of InterTrack, several requirements for the inter-domain traceback architecture are provided as follows:

- The architecture must leave each AS to decide to inherit a request of tracking by each AS's operational policy.
- The architecture should leave each AS to decide whether or not to investigate the inside of each own network domain more deeply. The architecture should also allow each sub-domain of an AS to decide whether or not to inspect each sub-domain's network by each sub-domain's operation policy.
- The architecture should allow each AS to take another action along with a tracking result such as a filtering or another tracking.
- The architecture should not forward request messages to ASes which have no relation to the issued attack.
- The architecture should not reveal sensitive information of an AS to others.
- A message exchanged in the architecture should have its own traceability to prove or to confirm the issuers of the message.
- The architecture should be independent from specific traceback techniques.
- The architecture should track back an attack on a dual stack environment, even when the attack employs some address translation techniques [74–76].

- The architecture should have the capability to cooperate with detection systems or protection systems.

- The architecture should exclude human beings as long as possible.

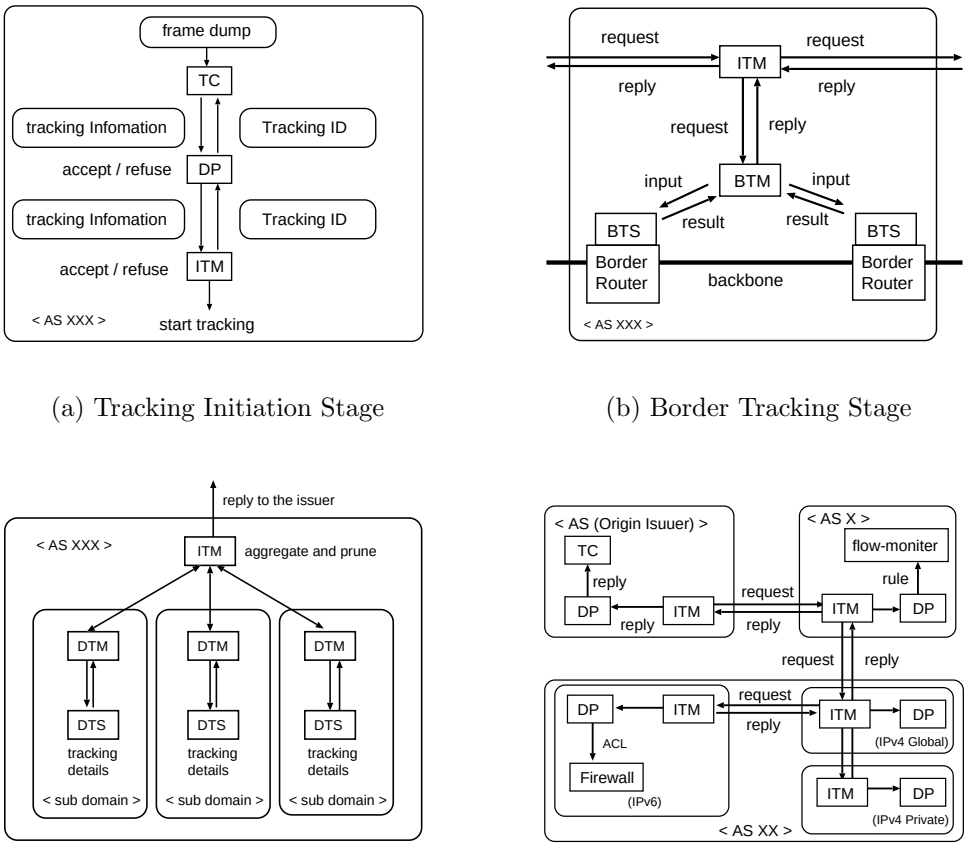
The first three requirements are essential for keeping the network operation boundaries. A routing operation reflects the contracts among ASes or the contracts between an AS and its customers. In order to keep such contracts and operational boundaries, network operators control each network domain with responsibility and cooperate with each other on the relationship of mutual trust based on such contracts. A traceback trail is a trial to confirm the routing path of unwanted traffic; thus, a traceback operation should follow the manner of other routing operation.

Next three requirements are used to block misuses of the traceback architecture. The operation of traceback will consume many resources on the related ASes; therefore, the traceback architecture should not generate or flood meaningless requests if possible. In order to reduce the damage of misuses, the message should not convey such sensitive information that might cause the leakage of secrets or confidence of an AS. Even when a misuse or a compromised action occurred, the traceability of the message will identify the offender.

The next two requirements deal with evasion attacks of each traceback technique. If the architecture depends on one specific traceback technique, attackers will develop evasion attacks and hide the location of the attacker nodes. In addition to this, many operation systems come to support the IPv4/IPv6 dual stack [77, 78], and several attacks come through a 6to4 IPv6 tunneling [79]. If the traceback architecture cannot track back attacks on the IPv6 network or attacks through some translators, the majority of attacks will shift in such a complex attack [79]. Hence, the independence from any specific traceback techniques and the independence of the versions of IP are required.

The last two requirements are supposed to automate the traceback operation. According to the taxonomy compiled by Mirkovic et al. [1], many DDoS attacks employ reflector nodes or step-stone nodes; therefore, the attacker nodes which are detected by a traceback trial might be just step-stones or reflectors. In order to detect commander nodes or true attacker nodes, the traceback architecture should cooperate with detection systems to start further traceback trials. As a matter of course, network operators will apply filters or run attack mitigation techniques after a traceback operation. To

automate the process of attack mitigation, the architecture should be able to export the result of a traceback trial as a trigger of the attack mitigation. Then, an attacker may change the pattern of attack traffic to avoid the effect of such mitigating actions. Combating with changes of a complex attack, the time spent to trace an attack path should be as short as possible. Because the time spent by a person is remarkably longer than the time spent by a computer, the architecture should exclude human beings if possible. It is a challenge to construct an automated traceback procedure while network operators on each AS can control the traceback operation on its own network according to each AS's operational policy.



(a) Tracking Initiation Stage

(b) Border Tracking Stage

(c) Intra-AS Tracking Stage

(d) Inter-AS Tracking Stage

Figure 3.1: Tracking on InterTrack

3.4 Overview of InterTrack

The main goal of InterTrack is to reconstruct the reverse AS path, which is the true attack path in AS hop level, and to detect the source ASes of an attack if possible. Another goal of InterTrack is to achieve the cooperation among traceback systems, detection systems and prevention systems inside an AS. InterTrack also aims to expedite the human operation and the cooperation among ASes on tracking an attack.

3.4.1 Architecture

The architecture of InterTrack refers to the Internet routing architecture. The reason why InterTrack refers to the Internet routing architecture is that the Internet routing architecture is designed along with the boundaries among operation domains and the differences of operational policies of each operation domains. For example, BGP shows the boundaries and contracts among ASes; on the other hand, OSPF sub-area can express the boundary of the different operation domain on an AS such as the boundaries among the network operation center and other departments in an enterprise network. Usually, a network operator cannot operate other network domains. On the network boundaries, network operators cooperate with each other to configure their own network equipments for achieving the proper routing. In a traceback trial, network operators cannot investigate other network domains as well as other network operations; therefore, they try to detect upstream neighbor ASes to ask them for further tracking.

In InterTrack architecture, each AS has a set of InterTrack components. A set of InterTrack components includes; the Inter-domain Tracking Manager (ITM), Border Tracking Manager (BTM), Domain Traceback Manager (DTM), Decision Point (DP), and Traceback Client (TC). Figure 3.1 shows the overview of InterTrack architecture. A phased-tracking approach is applied on inter-domain traceback trials through InterTrack. InterTrack separates a traceback trial in four stages along with network boundaries; *the tracking initiation stage* (Fig.3.1(a)), *the border tracking stage* (Fig.3.1(b)), *the intra-AS tracking stage* (Fig.3.1(c)) and *the inter-AS tracking stage* (Fig.3.1(d)). After accepting a traceback request on the tracking initiation stage, each AS preliminarily investigates its own status against the issued attack on the border tracking stage. On the border tracking, an AS judges by InterTrack whether or not the AS is suffered from an attack, whether or not the AS is forwarding malicious attack packets, or whether or not the AS is suspected of having attacker nodes on the inside. Triggered by the investigated AS status, InterTrack runs the inter-AS tracking stage and the intra-AS tracking stage in parallel.

3.4.2 Behaviors of InterTrack Components

An inter-domain traceback on InterTrack is composed of the federation of internal traceback trials on ASes through the phased-tracking stages. Here, we explain the

characteristics of each InterTrack components and behaviors of components on each tracking stage.

ITM on each AS controls traceback trials on its network domain along with the operation policy of the AS. ITM also mediates neighbor ASes to exchange the traceback information. Any traceback information from other ASes comes from the ITM network. The ITM network is an overlay network composed by a number of point-to-point peering of ITMs between two ASes. ITM network is mapped on the BGP-peering relationship so that each ITM communicates only ITMs on neighbor ASes according to the trust or the contract on the BGP-peering.

TC is an interface of InterTrack to request a traceback for network operators or detection systems such as intrusion detection systems. In the tracking initiation stage (Fig.3.1(a)), DP authenticates and authorizes TCs; on the other hand, ITM assigns a message ID to distinguish a traceback trial on the whole InterTrack architecture. DP is a separated module function of ITM to authenticate TCs and to control request rates from a TC. Because ITM not only deals with attack requests from the inside and also treat requests from neighbor ASes, the overhead of authentication and rate limits of TC may obstruct ITM in processing other requests. Therefore, the authentication function and the rate limit function of ITM are delegated to DP.

On the border tracking stage (Fig.3.1(b)), InterTrack preliminarily investigates the AS status expressed by the directions of the issued attack, by the possibility of the existence of an attacker on the inside of the AS, and by the information of address translation. When the result of the border tracking stage reveals the upstream neighbor ASes, InterTrack kicks off inter-AS tracking stage and propagates the traceback request message to each upstream neighbor AS through the ITM network. The neighbor ASes recursively runs the border tracking stage, and reconstruct the reverse AS path of the issued DDoS attack. Each AS adds its own AS status in the reply message and returns the reply message to the issuer neighbor AS through the ITM network. In parallel with the inter-AS traceback stage, an AS can start the intra-AS tracking stage when the result of the border tracking stage showed that the AS might have attacker nodes on the inside (Fig.3.1(d)). The intra-AS tracking stage is the deep internal inspection on each sub-domains of the AS (Fig.3.1(c)). The results of tracking on each sub-domain are aggregated by the ITM of the AS and stored in the AS's DP. The result of intra-AS tracking is not delivered to other ASes through InterTrack because the result of intra-AS tracking is an internal information and may include personal information such as

the information of the owner whose PC was infected by some worm.

On the border tracking stage and the intra-AS tracking stage, each AS can use various traceback techniques or traceback systems according to their characteristics. Border Tracking System (BTS) is a specified traceback system to investigate the direction of the issued DDoS attack on the network boundary and judge the AS status. On the other hand, Domain Tracking System (DTS) is a traceback system for deep internal inspection on an AS. In other words, a DTS is a traceback system to locate attacker nodes logically and physically (e.g., the logical location is the nearest edge router or the incoming port on a layer 2 switch; on the other hand, the physical location is the geographical location of the nearest router or switch).

Each traceback system communicates InterTrack through a BTM or a DTM. Both BTM and DTM are wrapper components to convert the traceback request to the input values for employed traceback systems, and translate the results of traceback systems into the InterTrack message. BTM and DTM can be implemented as a module or a proxy to exchange the traceback information with the manager server of a specific traceback system.

Each InterTrack component communicates with only neighbor components through an IPsec encrypted TCP connection [80] in order to reflect the trust among network domains properly, to protect abuses or attacks from unauthorized nodes, and to keep the information of a traceback secret from others. In addition, each ITM sends heartbeat messages to neighbor components in order to confirm the existence of each neighbor component and in order to adjust the time synchronization.

```
<?xml version="1.0"?>
<ITMTraceRequest>
  <DestinationITMID>v6-65002</DestinationITMID>
  <Origin>v4-65001</Origin>
  <SequenceNumber>10000</SequenceNumber>
  <TTL>5</TTL>
    <Footmark transform="yes">
      <PacketDump iftype="0x86">XXXX XXXX XXXX XXXX </PacketDump>
      <TimeStamp><sec>1132613480</sec><usec>159368</usec></TimeStamp>
      <TransPacket>
        <Border>6TO4</Border>
        <PacketDump iftype="0x86">XXXX XXXX XXXX XXXX </PacketDump>
      </TransPacket>
    </Footmark>
    <ITMPathList>
    <Origin>v4-65001</Origin>
    <NextHop depth="1">v4-65002</NextHop>
  </ITMPathList>
</ITMTraceRequest>
```

Figure 3.2: ITM trace request message

3.5 Reverse AS Path Reconstruction

In this section, we explain the detail of the reverse AS path reconstruction and discuss the feasibility of InterTrack. InterTrack reconstructs the reverse AS path of a DDoS attack through the border tracking stage and the inter-AS tracking stage. The border tracking stage on an AS reveals the AS status against the attack. If the border tracking stage judges that the AS received the attack traffic from some of upstream neighbor ASes, an ITM forwards an ITM trace request message to those upstream neighbor ASes which may forward the attack traffic. Here, we explain the detail of AS status and the consistency of ITM trace reply message which reveals the reverse AS path to the original issuer ITM. Figs. 3.2 and 3.3 show examples of an ITM trace request message and its ITM trace reply message.

3.5.1 AS Status against a DDoS Attack

An ITM decides actions by the AS status revealed on the border tracking. On the forwarding an ITM trace reply message, each ITM on the reverse AS path adds its AS status into the ITM trace reply message. The variations of AS status against a

```
<?xml version="1.0"?>
<ITMTraceReply>
  <SourceITMID>v4-65002</SourceITMID>
  <Origin>v4-65001</Origin>
  <SequenceNumber>10000</SequenceNumber>
  <TraceResult type="FOUND">
    <ITMSubTrees>
      <ITMSubTree depth="0" type="FOUND">
        <ITMID>v4-65002</ITMID>
        <NextHops>
          <Incomings>
            <ITMID>v6-65002</ITMID>
          </Incomings>
          <Outgoings>
            <ITMID>v4-65001</ITMID>
          </Outgoings>
        </NextHops>
      </ITMSubTree>
      <ITMSubTree depth="1" type="FOUND">
        <ITMID>v6-65002</ITMID>
        <NextHops>
          <Outgoings>
            <ITMID>v4-65002</ITMID>
          </Outgoings>
        </NextHops>
      </ITMSubTree>
    </ITMSubTrees>
  </TraceResult>
</ITMTraceReply>
```

Figure 3.3: ITM trace reply message

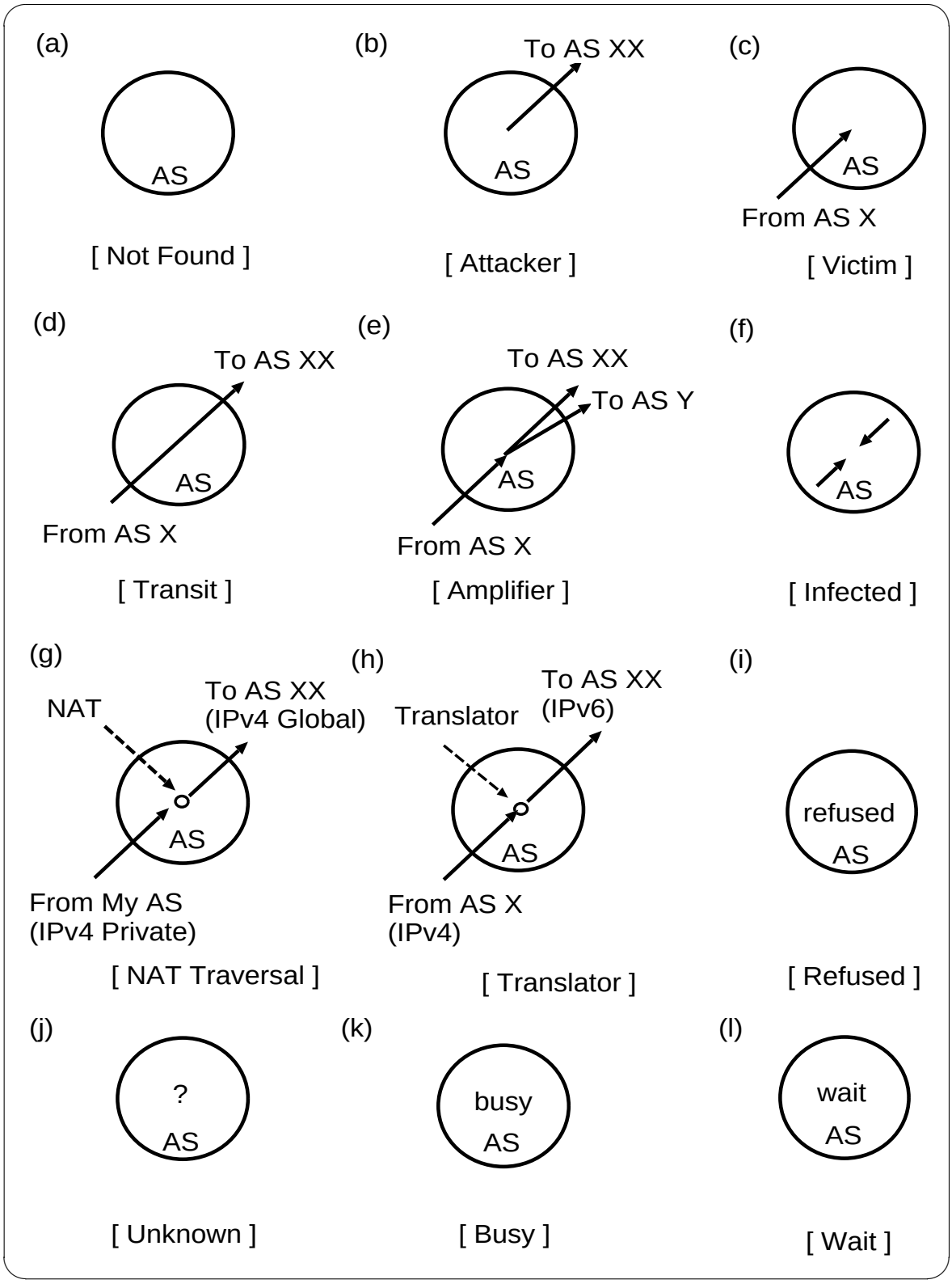


Figure 3.4: Variations of state of an AS on an attack

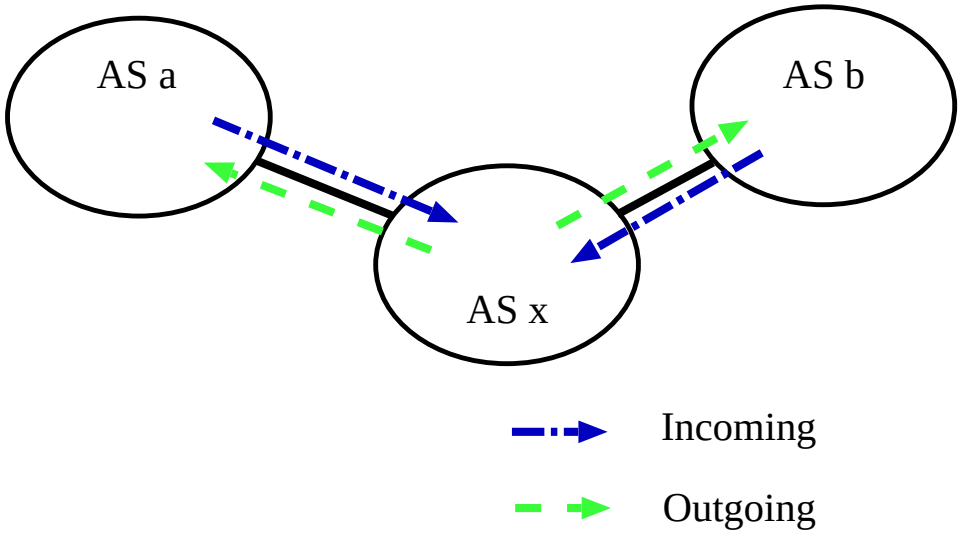


Figure 3.5: Directions of traffic on an AS

DDoS attack are shown in Fig. 3.4. On a traceback trial, an AS will have one of twelve variations of the AS status. Eight statuses can be expressed by the combination of following informations: directions of the forwarding path of an attack, the necessity of the detailed internal inspection, the notification of an address translation. In addition to these combinations, there are four error statues.

First, we consider the relation with a neighbor AS. Basically, an AS status is composed of the status of the inside and each status on the point to point (P2P)link between each neighbor AS. The status on a P2P link can be described by the packet directions, that is, *incoming* and *outgoing* (Fig. 3.5).

If an AS find a packet or flow of the issued DDoS attack on the incoming direction in a P2P link, the P2P link is a *Victim* link, that is, the AS receives the DDoS attack flow from the neighbor AS. The status of a P2P link is *Attack* when the DDoS attack is detected on the outgoing direction on the P2P link. When the DDoS attack is not detected either on the incoming direction or the outgoing direction of an P2P link, the AS status on the P2P link is *Negative*. If the DDoS attack is found both on incoming and on outgoing direction of a P2P link, it indicates a *Loop* has occurred. Usually, *Loop* is an error state which indicates the error of the BTS or some wrong routing so that a more detailed investigation is required. Therefore, a *Loop* link is defined as equal to the *Attack* and *Victim* link.

Next, we consider about the AS status with all relations between each neighbor AS. If all P2P links shows *Negative*, the AS status is judged as *Not Found* (Fig.3.4a). When each border tracking on each P2P link judges either *Negative* or *Victim*, an AS is in *Victim* state (Fig.3.4b). On the other hand, an AS has *Attacker* state when the results of P2P links either *Attack* or *Negative* (Fig.3.4c). An AS status is judged as *Transit* when the investigation results of P2P links contains *Negative*, *Attack*, and *Victim* (Fig.3.4d).

Here, we consider the AS status with the internal status of an AS. If there is a possibility that an attacker node is inside the AS, the AS is in *Amplifier* state (Fig.3.4e). The border tracking stage judges the AS status as an *Amplifier* state in the following cases:

- The number of *Attack* links is more than the number of *Victim* links.
- The amount of traffic to some *Attack* link is increased remarkably.
- One of P2P links shows *Loop*.
- When the AS connects to a neighbor AS with several links, one P2P link is *attack* and the other is *Victim*.

When an AS is in the *Amplifier* state or in the *Attacker* state, the AS has to start the intra-AS tracking. If there is a need to start intra-AS tracking, a BTM adds an *ASK_DTM* flag in the reply message to the ITM. A BTM also adds an *ASK_DTM* flag when the border tracking stage shows the *Infected* state (Fig.3.4f). The *Infected* state indicates that the AS is attacked not from other ASes, but from the inside.

Some attacks employ address translation techniques, such as IPv4/IPv6 tunneling or NAT [79]. If an attack comes from the IPv4 private segment which is operated by an AS, the border tracking judges the AS is in a *NAT traversal* state (Fig.3.4g). The border tracking stage indicates a *Translator* state when the attack comes through an 4to6 tunnel or 6to4 tunnel (Fig.3.4h). In these translation cases, a BTM adds the information of the translation. If an AS employs a logging technique such as SPIE [15] for the BTS, the BTS may store the previous packet information before the packet was translated. If BTS has the previous packet information, then the BTM also adds such previous packet information into the reply message. When an ITM receives the information of the translation from the BTM, the ITM adds the information into the

ITM trace request message. Then, in order to start further traceback trials in different address space, the ITM changes the role to the ITM on another address space, or forwards ITM trace request message to another ITM on the same AS.

Finally, we consider four error states on the border tracking. Each AS can refuse a traceback request along with its operational policy. If an AS refuses a traceback request, then, the ITM adds a *Refused* state as its AS status into the ITM trace reply message, and sends the ITM trace reply message to the issuer neighbor ITM (Fig.3.4i). If an ITM, a BTM, or a BTS is busy because of processing other traceback requests, the AS status becomes *Busy* (Fig.3.4j). When something wrong has occurred in the border tracking stage, The BTM replies with some error message. Then, the AS status is judged as *Unknown*, the ITM adds an error message from the BTM into the ITM trace reply message (Fig.3.4k). An ITM will reply *Wait* as the AS status if an AS is in the border tracking stage, but the AS needs much more time to get the result due to the limitation of the BTS (Fig.3.4l).

3.5.2 Loop Detection on Forwarding an ITM Trace Request Message

In order to avoid the loop in forwarding an ITM trace request message, the ITM on each adds its *ITM ID* into the *ITM Path List* field on the ITM trace request message when the ITM forwards the ITM trace request message to neighbor ASes. Using the ITM path list and the message ID of an ITM trace request message, each ITM judges whether a loop on the message forwarding occurs or not.

An ITM has ITM IDs which represent the address spaces covered by the ITM. The ITM ID on each address space is represented by the address space information and AS number. For example, each ITM ID of AS 2500 is `v4-2500` in the IPv4 global network, `v4-2500-private` in the IPv4 private network, `v6-2500` in the IPv6 network. The origin ITM, which originates an ITM trace request message, assigns a message ID to a traceback request on the tracking initiation stage. A message ID is composed of the ITM ID of the origin ITM and the sequence number assigned by the origin ITM.

When a result of the border tracking contains a neighbor AS as the upstream neighbor and the ITM ID of the neighbor AS is already included in the ITM path list field, then, an ITM concludes a loop occurs and the ITM does not forward the ITM trace request to the neighbor ITM. An ITM also judges a loop when the receive ITM

trace request message contains its own ITM ID in the ITM path list field. In this case, the ITM does not reply with the ITM trace reply message to the issuer neighbor ITM. The ITM path list field contains all ITM IDs which represent the partial reverse AS path of the issued attack. Each ITM ID indicates each traversal AS of the ITM trace request; therefore, an AS can refuse the traceback request when an untrusted AS is included in the ITM path list.

In addition to the ITM path list field, an ITM trace request message contains Time-To-Live (TTL) field in order to stop an endless forwarding. Each ITM decrements the value of the TTL field on forwarding an ITM trace request message. If the TTL value reaches zero or a negative value, an ITM stops forwarding the ITM trace request. According to the analysis by team Cymru [81] or by Geoff Huston [82], the observed maximum AS path length was less than 40 hops, and the weekly average AS hops is about 5 hops. Therefore, the maximum TTL is settled in 64 with some provisioning and the default TTL value is defined as 5.

3.5.3 Inconsistency among Tracking Results of each AS

An ITM trace reply message may contain some inconsistencies among each ITM result on the Reverse AS path. The inconsistency will occur in these cases as follows: When the ITM trace reply message reports *Victim* or *Infected* state on the result of other ITM, this inconsistency is caused by two kinds of mistakes. One case is that the BTM, whose AS is in several hops away from the origin ITM, mis-judged the AS status as *Victim* or *Infected*. The other inconsistency is that an ITM request message was generated and forwarded from a transit AS and the ITM request message wrongly reached the true *Victim* AS by the mis-judges in several ASes.

If the ITM trace reply message contains *Not Found* on the results of other ASes, an ITM made an incorrect forwarding to neighbors that are not in the real attack path, or the BTM of an AS mis-judged the AS state as *Not Found*. The ITM Subtree does not have the ITM ID of the former issuer on the outgoings field when the result of the border tracking on the deeper hop AS was wrong, or when the mis-forwarding continuously occurred in several hops.

The ITM trace reply message may contain a loop in the *ITM Subtrees* field. If the same ITM ID is listed both on the Outgoings filed and the Incomings field of a ITM Subtree field, the ITM Subtree would indicate that the border tracking on the AS was

wrong or the routing loop really occurred between two ASes. When a shallower hop ITM's ID is contained in the Incomings field on the result by the deeper hop ITM, the BTM on the deeper hop AS mis-judged, or the wrong message forwarding continuously occurred from the shallower hop AS to the deeper hop AS.

Even when an ITM trace reply message contains some inconsistency, the network operator on the original issuer AS can contact each AS by referring to the reverse AS path on the ITM trace reply message. On the other hand, each AS on the reverse AS path holds the partial ITM paths both on the received ITM trace request messages and on the ITM trace reply messages from neighbor ITMs. Hence, each AS can confirm the inconsistency on the reverse AS path by itself even when the AS is not the original issuer AS.

3.5.4 Analysis of Attack Cases against the InterTrack

Attack cases to the InterTrack architecture are considered here, and the feasibility of InterTrack against each attack case is then discussed, while the defending techniques to cover the vulnerabilities of InterTrack are explored.

Against the Numerous requests from a TC, DP can limit or drop requests from TC in a unit time. In addition to this, DP's authentication and rate-limit functions can be separated from ITM; therefore, the ITM network cannot be affected by the processing rate-limits on DP and can treat other requests while a TC attacks by numerous requests.

An attacker may try to hijack a TCP connection between components. In InterTrack, each TCP session between two components is based on IPSec authentication [80]; therefore, each component will not accept the request of connection from an unauthorized node. Moreover, network operators can easily apply filter rules on the nearest routers for each component in the source address and the destination address pair, because the neighbor components of each component are fixed or limited. Also, the IP address of a component should be known by only neighbor components, hence, it is not necessary to assign a FQDN to the IP address of a component and to propagate the FQDN by DNS. Therefore, network operators can hide the IP addresses of InterTrack components from unauthorized people. Furthermore, if an ITM has several interfaces, each connection to a neighbor component can be achieved on a closed private network. By a combination of such techniques, network operators can protect

InterTrack from SYN floods, or UDP floods, even when the attacker spoofs the source IP address of attack packets as the IP address of a component.

Since each connection of two components can be constructed on the closed private network, such a private network is not affected by the bandwidth consumption on the main link of a neighbor AS. Even when the connection of two components is achieved as the on-line connection, priority queuing techniques can turn down the effect of the bandwidth consumption against the InterTrack messaging. Also, the ITM network is an overlay network. If other routing paths are prepared, the ITM network can tolerate the influence of link-downs or route flaps.

If an ITM is hijacked by an attacker, it becomes a serious security issue over several ASes because the attacker can steal the inside information of the hijacked ITM's AS and can attack by faked ITM trace request messages and spoofed ITM race reply messages. By hijacking other InterTrack components, an attacker can steal the inside information or can dirty traceback results. Therefore, each AS should protect the intrusion to the InterTrack as strongly as possible.

When an ITM is hijacked by an attacker, the hijacked ITM may send a faked ITM trace reply message which contains a spoofed reverse AS path. In this case, if the network operator on the original issuer AS contacts all ASes on the spoofed reverse AS path to verify the reverse attack path, then, the network operator on each AS will find the inconsistency on the same traceback trial and will detect the hijacked AS.

3.6 Discussion

3.6.1 A Multi-Layer Traceback for Complex Attacks

Considering DDoS tools [1], the attacker nodes on the Attacker AS are just the step-stone nodes or the reflector nodes. In order to detect commander nodes or the PC of an attacker, InterTrack tracks the attack in multi-layers by the cooperation of detections systems and continuous trials of traceback. After the intra-AS tracking stage (Fig.3.1(c)), the result of the intra-AS tracking is stored in DP. According to the configuration or AS's policy, DP exports the result to detection systems to correct the command packets or the pre-reflected packets (Fig.3.1(d)). If a detection system catches these command packets or pre-reflected packets, it starts another traceback to the source of these packets. Proceeding this process recursively, InterTrack can track an attack on the layer 7 networks, that is, traces back the attack from the step-stone node to the *the true source* of the attack.

This continuous traceback trial is used to track the attacker node on the inside of an AS. When an *Infected* status AS requests a traceback, the ITM of the AS starts the intra-domain tracking stage soon after. The purpose of the intra-AS tracking stage is not only revealing the reverse path on the layer 3 network but also reconstructing a reverse path on a layer 2 network and locating the attacker node. There are several traceback techniques on a layer 2 network [18, 22, 23, 38, 83], and most of them require not the IP datagram of the issued attack but the Ethernet header to get the source MAC address or VLAN information as a key of tracing. In order to track on a layer 2 network in the *Infected* case, we design the *PacketDump* field of the trace request message to include the packet payload with the Ethernet header. The process of the traceback on layer networks as follows: First, the operator or a detection system sends a whole Ethernet frame to InterTrack through a TC. After judged *Infected*, the ITM kicks the intra-AS tracking stage and finds the nearest layer 3 gateway of the attacker node. If the attacker node is in the same layer 2 network of the victim node, then DTM runs a layer 2 traceback technique and tracks the location of the attacker node. When the attacker node is in another local subnet, the DTM explores the layer 2 subnet to detect the logical and physical location of the attacker node if the DTS on the attacker's subnet stores the source MAC address of the issued packet. If not, the DP of the AS exports the result of the initial traceback to the detection system on

the attacker's subnet. And then, the detection system triggers another tracking by catching an Ethernet frame of the issued attack to get the source MAC address.

3.6.2 Privacy Issues

In order to start a traceback trial, a TC has to input the whole Ethernet frame of the issued attack packet. Then, the privacy issue about the packet payload comes into question. Focusing on the DDoS attack, the packet payload is usually meaningless information or the binary of a malicious code; therefore, such packets do not include any personal information. A single packet attack such as SQL slammer does not have any personal information either. Hence, the privacy issue of the packet payload of a trace request will not matter as long as InterTrack is employed for the traceback of attacks. Of course, if someone tries to track other service traffic by InterTrack, the privacy issue comes to a head.

InterTrack can trace an attack in the multi-layers. Running a multi-layer traceback requires some auditing systems which store the layer 7 / application level audit log or bind a MAC address to an IP datagram as its source. Using such audit systems, privacy issues of the audit log must be considered.

3.6.3 Certification on InterTrack Components

Each component of InterTrack achieves IPSec encapsulated TCP sessions [80] with all neighbor components; consequently,the overhead of exchange shared keys or certificate files among ASes must be considered. The APNIC has a trial of certification of IP addresses and ASes [84] by X.509 Extensions for IP Addresses and AS Identifiers [85]. These X.509 Extensions are expected to feed into sBGP [86] or soBGP [87]. The certificate files for sBGP or soBGP can be used for the IPsec Encapsulating Security Payload (ESP) [80] between each neighbor InterTrack components as well as the IPSec ESP among components of SPIE [15].

The prototype design of the ITM trace request message shown in Fig. 3.2 is simple and it does not include the digital signatures of traversed ASes. The digital signature is effective for proving or verifying the issuer AS. The InterTrack messages are designed in XML format; therefore, an extension of messages to include such digital signature can be easily developed by adding a sub-element in the ITM path list field. The extension

for the digital signatures is part of future research with the public key sharing problem mentioned above.

3.7 A Prototype Implementation of InterTrack

This section shows the detail of the prototype implementation of InterTrack. The basic functions of InterTrack have been developed in C language on FreeBSD, except for the function applying the AS's operational policy and IPSec encryption. LIBXML2 [88] was employed for the XML parser of InterTrack messages. We also developed a sample BTM implementation and a sample DTM implementation as a proxy for PAFFI [16]. PAFFI [16] is an implementation of hash-digest-logging traceback method by Yokogawa Electric Corporation. The volume of the prototype code was about 50,000 lines totally.

3.7.1 Library and InterTrack Components

We developed a library which contains common APIs used by each InterTrack components. Fig. 3.6 shows the software architecture of InterTrack. Basically, each InterTrack component processes XML messages; therefore, each component uses common APIs on message processing, on reading or writing messages, on serializing InterTrack messages, on exchanging heartbeat messages, or on managing connections between neighbor nodes. In this software architecture, each InterTrack component is modularized. Each component can be implemented by changing the algorithm of the traceback module. The characteristics of each part in Fig. 3.6 are as follows:

- *Connection*
  This part manages each connection between neighbor nodes. According to the configuration file or changes through command line, ConnectionManager connects or accepts a TCP session to a neighbor node. The DispatchLoop pools file descriptors or sockets and calls ReaderWriter to process each buffer stream or message. ThreadManager controls the mechanism or policy on the DipatchLoop. We have developed only select-based DispatchLoop, however, a DispatchLoop implementation using thread could be easily implemented in the ThreadManager.

- *ReaderWriter*
  ReaderWriter controls network I/O or file I/O. Through Message Processing APIs, ReaderWriter passes InterTrack messages to MessageDispather or writes InterTrack messages into socket file descriptors.

- *Message Processing API*
  We prepared several APIs to manipulate InterTrack messages. XMLObjectMapping maps an InterTrack message in an XML string to an internal structure for InterTrack messages. We defined WITMSG as the internal structure for InterTrack messages shown as Fig 3.7. Message Management API is a group of APIs to manipulate an InterTrack message on WITMSG structure.

- *Message Dispatcher*
  Message Dispatcher dispatches or discards messages according to the types of messages and the type of traceback module.

- *NeighborNode*
  NeighborMannager manages the entries of the NeighborTable. The Neighbor Table contains the information neighbor nodes which are defined by configuration files or new configuration through command lines. The Traceback module or the Heartbeat module looks up the information of neighbor nodes through APIs of the NeighborManager.

- *TraceEntry*
  TraceEntry is a table used to arrange several InterTrack messages in one traceback trial. The Traceback module registers a new InterTrack message, looks up an existing TraceEntry, or removes an old TraceEntry through the TraceEntryManager.

- *Module*
  The algorithm parts of each InterTrack component were modularized. When developing a BTM or a DTM, developers make the algorithm and combine the library of a specific traceback implementation on this part.

- *bootstrap*
  The Bootstrap initializes each part and starts an InterTrack component as a daemon.

Each InterTrack component was developed as a daemon; *itmd*, *btmd*, *dtmd*, *dpointd*, and *witclient*. As a sample TC, *packetcapture* was developed, which captures packets by PCAP library and passes the captured packets to *witclient* through a ring buffer on the shared memory.

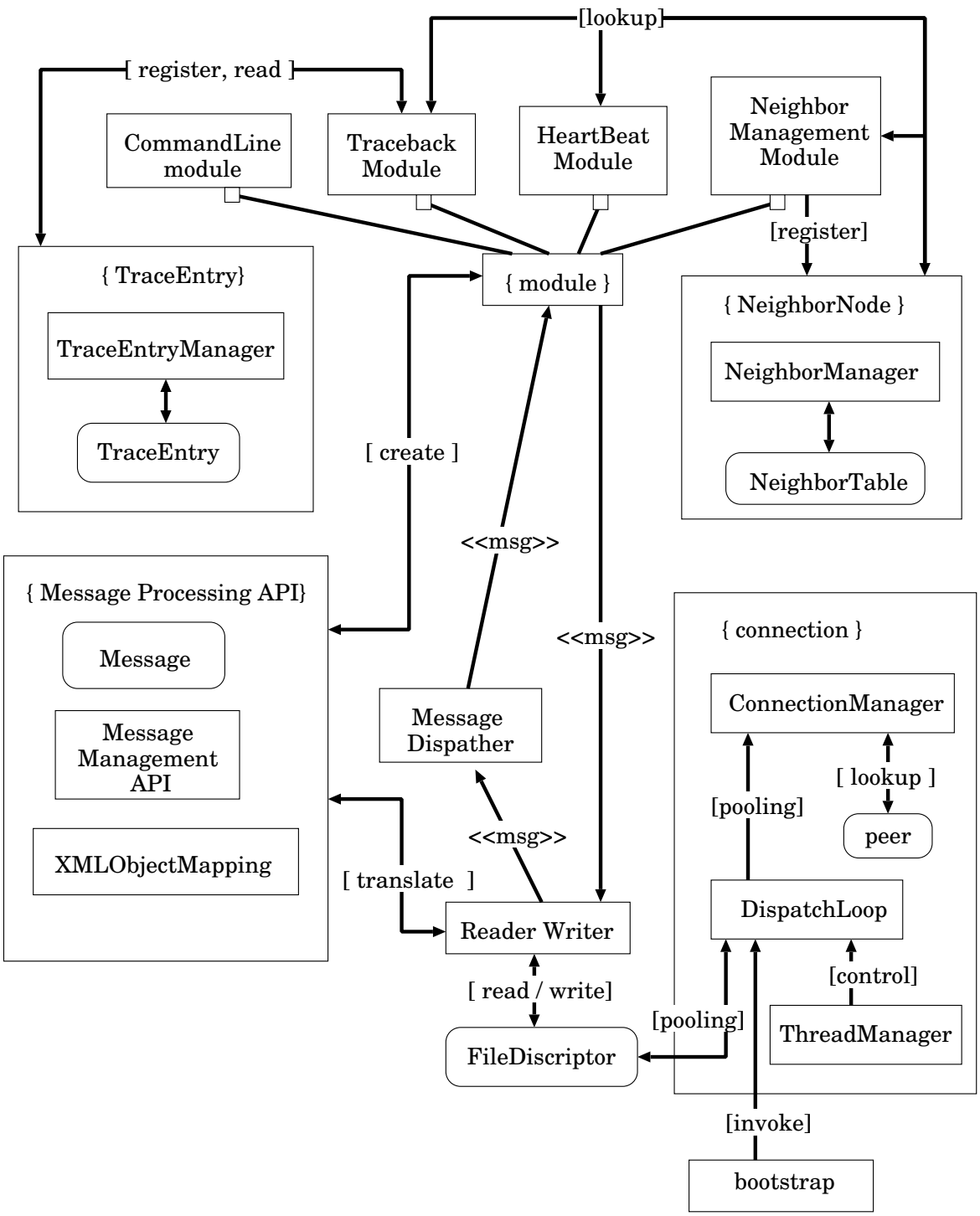


Figure 3.6: The software architecture of InterTrack

```
typedef struct witmsg{
    MSG_TYPE msg_type;
    union{
        heartbeatReq    *hbreq;
        heartbeatRep    *hbrep;
        heartbeatErr    *hberr;
        clientTraceReq  *clientTraceReq;
        clientSeqRep    *clientSeqRep;
        clientTraceRep  *clientTraceRep;
        dpointTraceReq  *dpointTraceReq;
        dpointSeqRep    *dpointSeqRep;
        dpointTraceRep  *dpointTraceRep;
        itmTraceReq     *itmTraceReq;
        itmTraceRep     *itmTraceRep;
        btmTraceReq     *btmTraceReq;
        btmTraceRep     *btmTraceRep;
        dtmTraceReq     *dtmTraceReq;
        dtmTraceRep     *dtmTraceRep;
    } msg;
}WITMSG;
```

Figure 3.7: WITMSG structure

3.7.2 Sample BTM and DTM Using PAFFI

Besides the library and daemons, a sample BTM implementation and a sample DTM implementation using PAFFI [16] were developed. The PAFFI architecture contains one manager node (PAFFI Manager) and several capture point nodes (Footmakers) which record captured packets in a Bloom filter [40]. Each Footmarker has several Bloom filters. Each Bloom filter, called a *capture point*, is mapped with an interface or a MAC address filter rule; therefore, a Footmarker can distinguish incoming traffic and outgoing traffic according to the identifier of each capture point. Hence, PAFFI can be used as BTS which can reply the variations of AS status described in section 3.5.1. Fig 3.8 shows a sample topology when PAFFI is used as BTS.

In sample BTM/DTM implementations for PAFFI, we used the proxy type implementation style, that is, both BTM and DTM behave as clients of a PAFFI manager and translate from an InterTrack trace request message to the PAFFI request message or a PAFFI reply message to an InterTrack trace reply message. The messages between the PAFFI manager and its client are described in XML, and client sends and receives messages over HTTP [89]. Unfortunately, a PAFFI reply message does

Table 3.1: AS mapping table on BTM for PAFFI in accordance with Fig. 3.8

Capture Point ID	AS number	direction
capture point 1	Y	incoming
capture point 2	Y	outgoing
capture point 3	Z	incoming
capture point 4	Z	outgoing

not contain AS-number field in order to indicate an upstream neighbor AS; therefore, we prepared AS mapping table on the BTM implementation. The components of the AS mapping table can be described in Fig 3.1. Using this AS mapping table, BTM converts a capture point ID to the AS number of a neighbor AS and the direction of the issued packet.

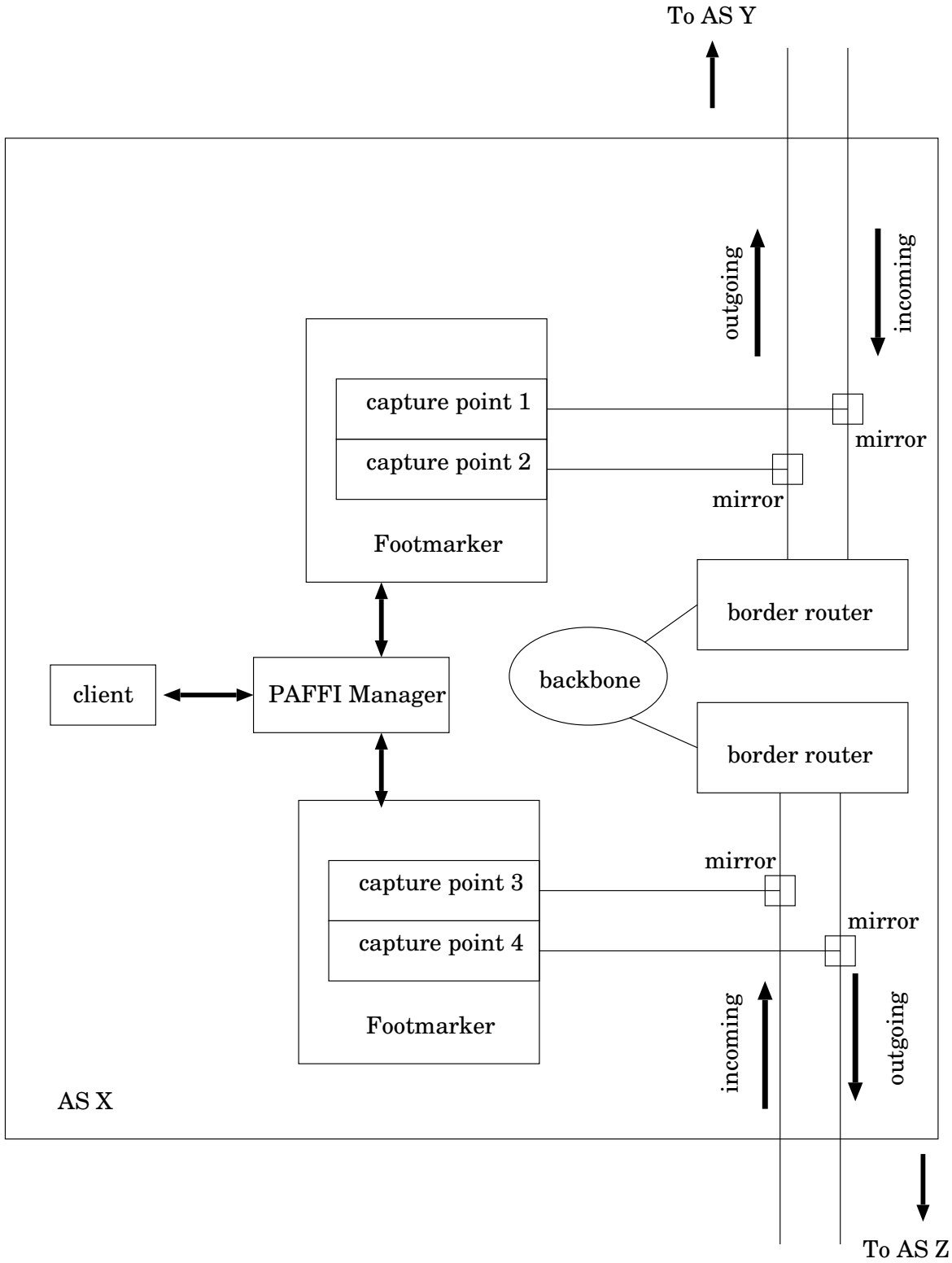


Figure 3.8: The topology of PAFFI as BTS

3.8 Preliminary Evaluation

3.8.1 Expected Round Trip Time of an ITM Trace Request

With InterTrack, users will wonder how much time will be spent to get a reverse AS path. Here, we consider the round trip time (RTT) of an ITM trace request. For the sake of simplicity, we assumed that each AS employs a hash-digest-logging method as the BTS on each network. Because of the characteristic of hash-digest-logging method, the reverse AS path always becomes a liner topology.

If an ITM trace request message is forwarded from Origin issuer AS to the AS in n hops, and the AS in n hops is the Attack state, suppose each parameters as follows:

- $t_{dec(i)}$: the time from when AS i received an ITM trace request to when the AS i decides to start the border tracking stage or to refuse the request.
- $t_{btm(i)}$: the time spent for border tracking stage
- $t_{req(i)}$: the time spent to forward an ITM request message to neighbor ASes
- $t_{wait(i)}$: the time spent to wait for all neighbor ASes to return each ITM reply message
- $t_{rep(i)}$: the time spent to make and send an ITM trace reply message
- $t_{out(i)}$: the time out threshold of $t_{wait(i)}$
- $t_{rtt(i)}$: the RTT of a traceback, that is, from the time when AS i receives an ITM trace request to the time when the AS finished sending the corresponding ITM trace reply message.

Then, the maximum RTT and the minimum RTT on the AS i are

$$t_{max(i)} = t_{dec(i)} + t_{btm(i)} + t_{req(i)} + t_{out(i)} \tag{3.1}$$

$$t_{min(i)} = t_{dec(i)} + t_{rep(i)} \tag{3.2}$$

$$t_{min(i)} \leq t_{rtt(i)} \leq t_{max(i)} \tag{3.3}$$

$$t_{rtt(i)} = t_{dec(i)} + t_{btm(i)} + t_{req(i)} + t_{wait(i)} + t_{rep(i)} \tag{3.4}$$

$$\tag{3.5}$$

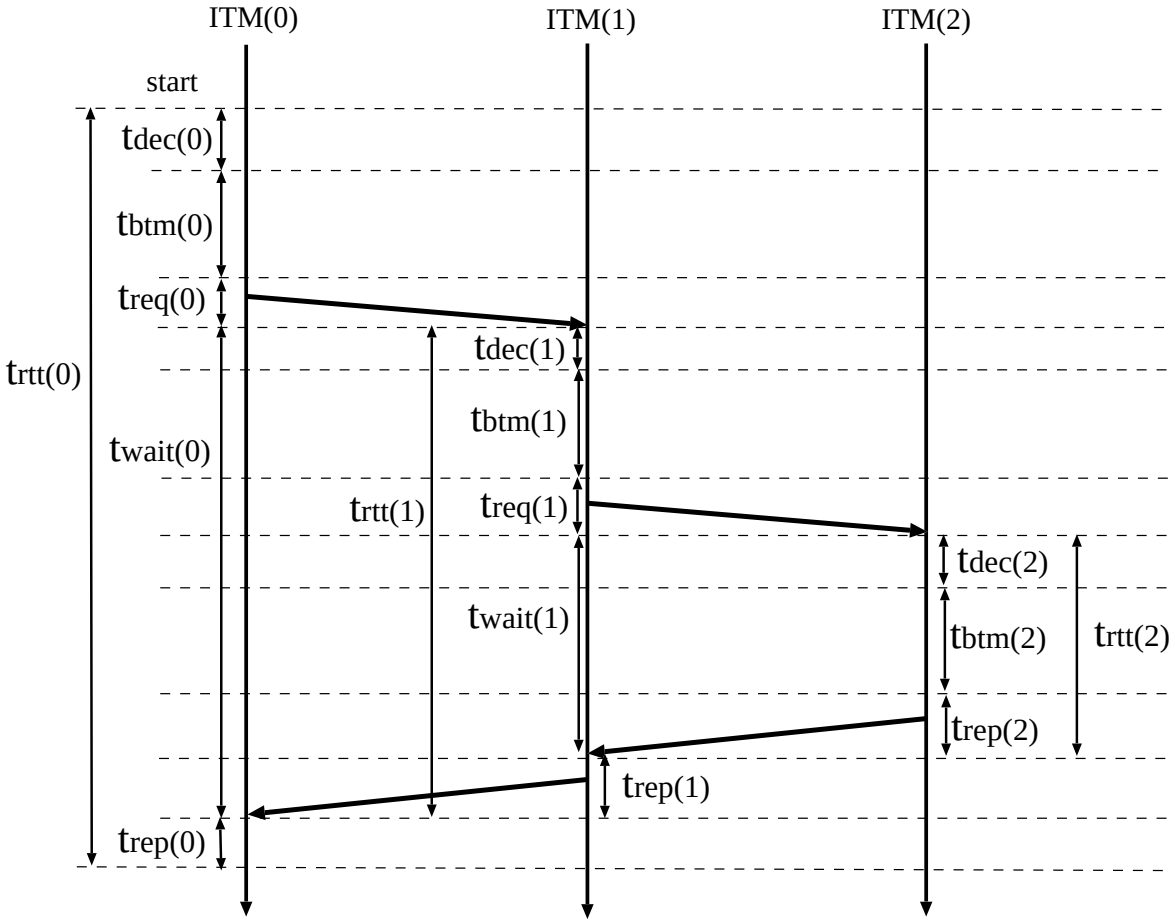


Figure 3.9: The round trip time of a response of an ITM trace request with 3 ASes

On the original issuer AS ($i = 0$), the RTT is

$$t_{rtt(0)} = \sum_{0}^{n}(t_{dec(i)} + t_{btm(i)} + t_{req(i)} + t_{rep(i)}) \quad (where\ t_{req(n)} = 0) \tag{3.6}$$

Next, we consider the expectation of RTT on AS i. Suppose the probabilities on each decision of AS i as follows:

- $p_{dec(i)}$: the probability with which AS i decides to start the border tracking stage.
- $p_{req(i)}$: the probability with which AS i decides to forward the ITM trace request to upstream neighbor ASes as the result of the border tracking stage.
- $p_{out(i)}$: the probability with which the time out on the waiting ITM trace reply messages occurs.

Here, the probability that AS i receives an ITM trace request messages from all neighbor ASes is described as :

$$p_{wait(i)} = p_{dec(i)}p_{req(i)}(1 - p_{out(i)}) \tag{3.7}$$

The expectation of RTT in the original issuer AS ($i = 0$) is

$$E(t_{rtt(0)}) = \sum_{i=0}^{n}(t_{min(i)} + t_{btm(i)}p_{dec(i)} + (t_{req(i)} + t_{out(i)})p_{dec(i)}p_{req(i)})(\Pi_{k=0,i}p_{wait(k)}) \quad (where\ t_{req(n)} = 0) \tag{3.8}$$

Next, we consider the RTT with the false positive rate and the false negative rate on the border tracking stage. The probabilities of mis-detection of the border tracking stage are defined as follows:

1. $p_{bfp(i)}$: the false positive rate of the border tracking stage. Here, AS i does not have an upstream neighbor AS on the attack path, but the border tracking stage mis-judges and indicates an upstream AS.

2. $p_{bfn(i)}$: the false negative rate of the border tracking stage. Here, AS i has an upstream neighbor AS on the attack path, but the border tracking stage mis-judges and does not indicate an upstream AS.

3. $p_{btp(i)}$: the true positive rate of the border tracking stage. Here, AS i has an upstream neighbor AS on the attackpath, and the border tracking stage properly judges and indicates an upstream AS.

4. $p_{btn(i)}$: the true negative rate of the border tracking stage. Here, AS i does not have an upstream neighbor AS in the attack path, and the border tracking stage judges properly and does not indicate an upstream AS.

$$p_{btp(i)} + p_{btp(i)} + p_{bfp(i)} + p_{bfn(i)} = 1 \tag{3.9}$$

$$p_{req(i)} = p_{bfp(i)} + p_{btp(i)} \tag{3.10}$$

$$p_{fwd(i)} = p_{dec(i)} p_{req(i)} = p_{dec(i)}(p_{bfp(i)} + p_{btp(i)}) \tag{3.11}$$

$$p_{wait(i)} = p_{dec(i)}(p_{bfp(i)} + p_{btp(i)})(1 - p_{out(i)}) \tag{3.12}$$

$$p_{err(i)} = p_{dec(i)}(p_{bfp(i)} + p_{btp(i)}) p_{out(i)} \tag{3.13}$$

$$E(t_{rtt(0)}) = \sum_{i=0}^{n} (t_{min(i)} + t_{btm(i)} p_{dec(i)} + (t_{req(i)} + t_{out(i)}) p_{dec(i)}(p_{bfp(i)} + p_{btp(i)}))(\Pi_{k=0,i} p_{wait(k)})$$
$$(where\ \ p_{wait(n)} = 0,\ \ p_{req(n)} = 0) \tag{3.14}$$

3.8.2 Preliminary Experiments with Implementation

To evaluate the basic workloads of the ITM network, a preliminary experiment was conducted with the prototype implementation of InterTrack. For the experimental environment, 9 Dell Power Edge 1855 blade servers were used as a testbed network. The physical topology is shown in Fig. 3.10. Each blade server equipped with a Pentium III 1.4GHz CPU, a 1024 MB RAM, and two gigabit Ethernet interfaces. These 9 blade servers were divided into two blade cages and interconnected with each other through the two layer 2 switches equipped on each blade cage. In this experiment, we measured the overhead on the message processing of ITM. A blade server was regarded as an AS,

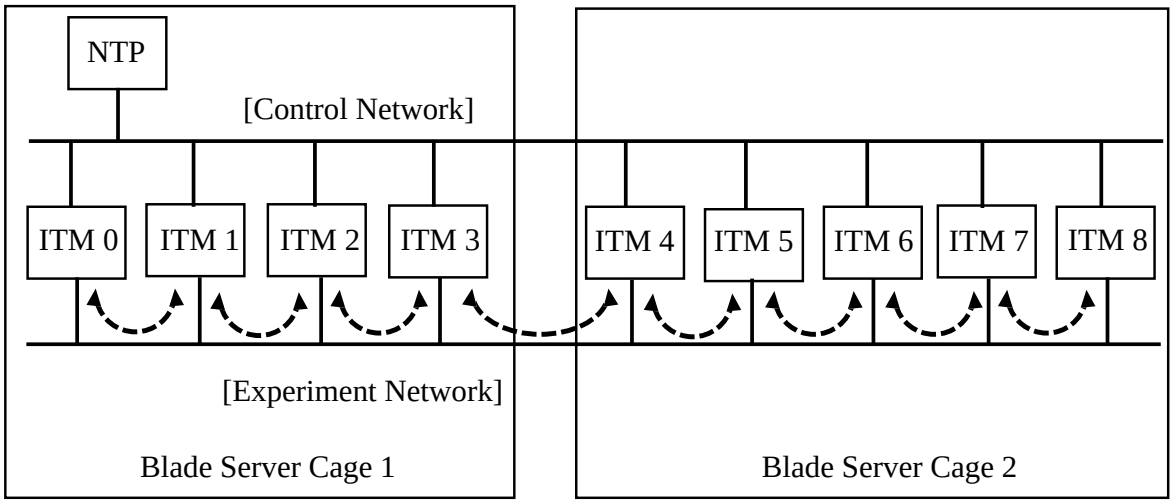


Figure 3.10: Testbed topology

an ITM on each blade server was run with a dummy BTM function which judged all neighbor ITMs except the issuer one as upstream neighbors. On the *Control Network*, each ITM synchronized its own time to the NTP server and other ITM nodes. On the *Experiment Network*, each ITM connected to only neighbor ITMs in a liner topology according to the assumption that each AS employed a hash-digest-logging method. We measured the relation between the RTT of ITM trace request messages on the origin issuer ITM and the number of hops forwarding the ITM trace request messages. With over-provisioning about the average length of AS hops [81, 82], we measured RTTs of the 1,000 ITM trace requests which were forwarded from 1 hop to 8 hops. Each trial ran independently with 5 second intervals. The average RTT of ICMP on the *Experiment Network* between each neighbor node was 0.197 milliseconds.

Fig. 3.11 shows the box-whisker plot of the experiments. Including outliers, all messages returned to the original issuer ITM within 1.2 seconds, and most were less than 200 milliseconds. Table 3.2 shows the data of the components on Fig. 3.11, Fig 3.12, and Fig. 3.13. Because the value of the mean is higher than the 90th percentile on each column, each column of Fig. 3.11 draws a positive skew like Fig 3.12.

In order to analyze the trend of the curve more deeply, the variations of RTT which were less than 15 milliseconds were plotted in Fig. 3.13. The distribution of RTT on each hop length was folded in a narrow box, but the outliers of each column make the variance high. Fig. 3.13 focused on the values between the 10th percentile and the

Table 3.2: The data of box-whisker plot on Fig. 3.11 and Fig .3.13

	1	2	3	4	5	6	7	8
max. (msec.)	199.939	199.956	1202.139	200.032	105.224	199.9	200.141	199.968
mean (msec.)	10.400	8.7806	13.155	12.604	12.124	13.348	14.903	14.887
90th percentile (msec.)	1.0180	2.0300	3.173	4.1960	5.4410	6.9200	8.1240	9.6870
75th percentile (msec.)	0.8125	1.7215	2.717	3.7720	4.8670	6.1725	7.4320	8.8835
median (msec.)	0.7025	1.5890	2.542	3.5815	4.6245	5.8940	7.1310	8.5300
25th percentile (msec.)	0.6750	1.5120	2.406	3.4265	4.4650	5.6530	6.9295	8.2650
min. (msec.)	0.5330	1.3360	2.131	2.9930	4.0200	5.1790	6.4980	7.7410
variance	931.24	679.67	4891.3	831.22	672.28	689.99	715.65	571.68

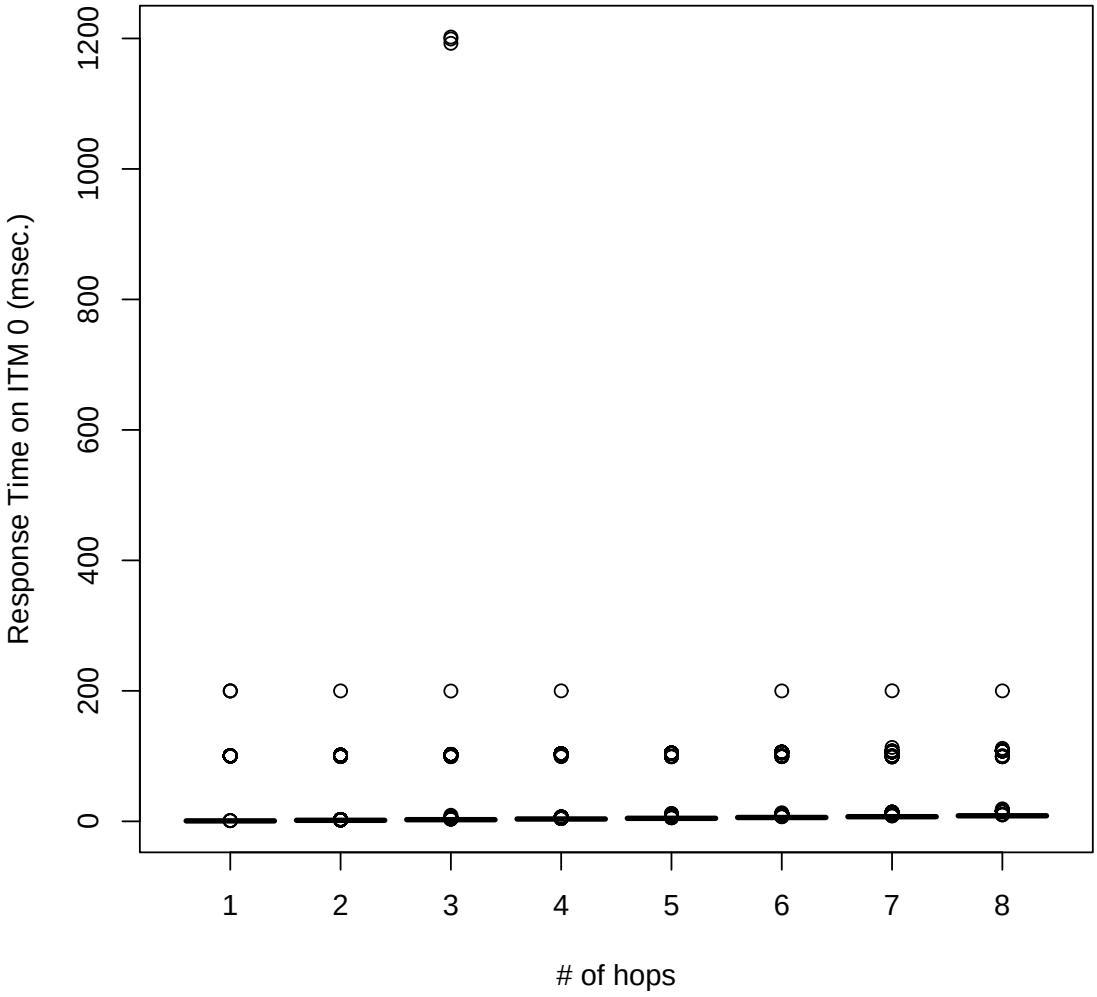


Figure 3.11: RTT of a ITM trace request in a liner topology

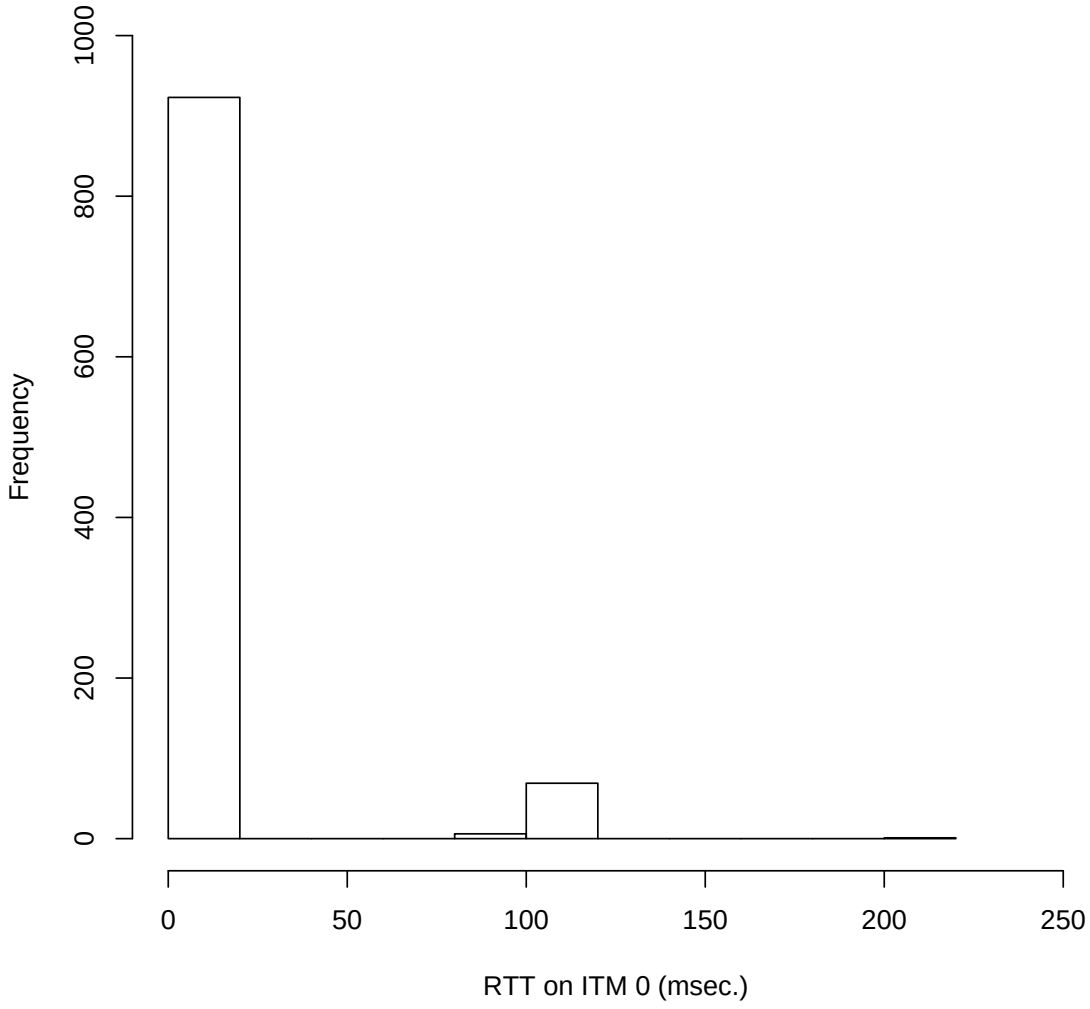


Figure 3.12: Histogram of RTT on ITM 0 in a 9hops length topology

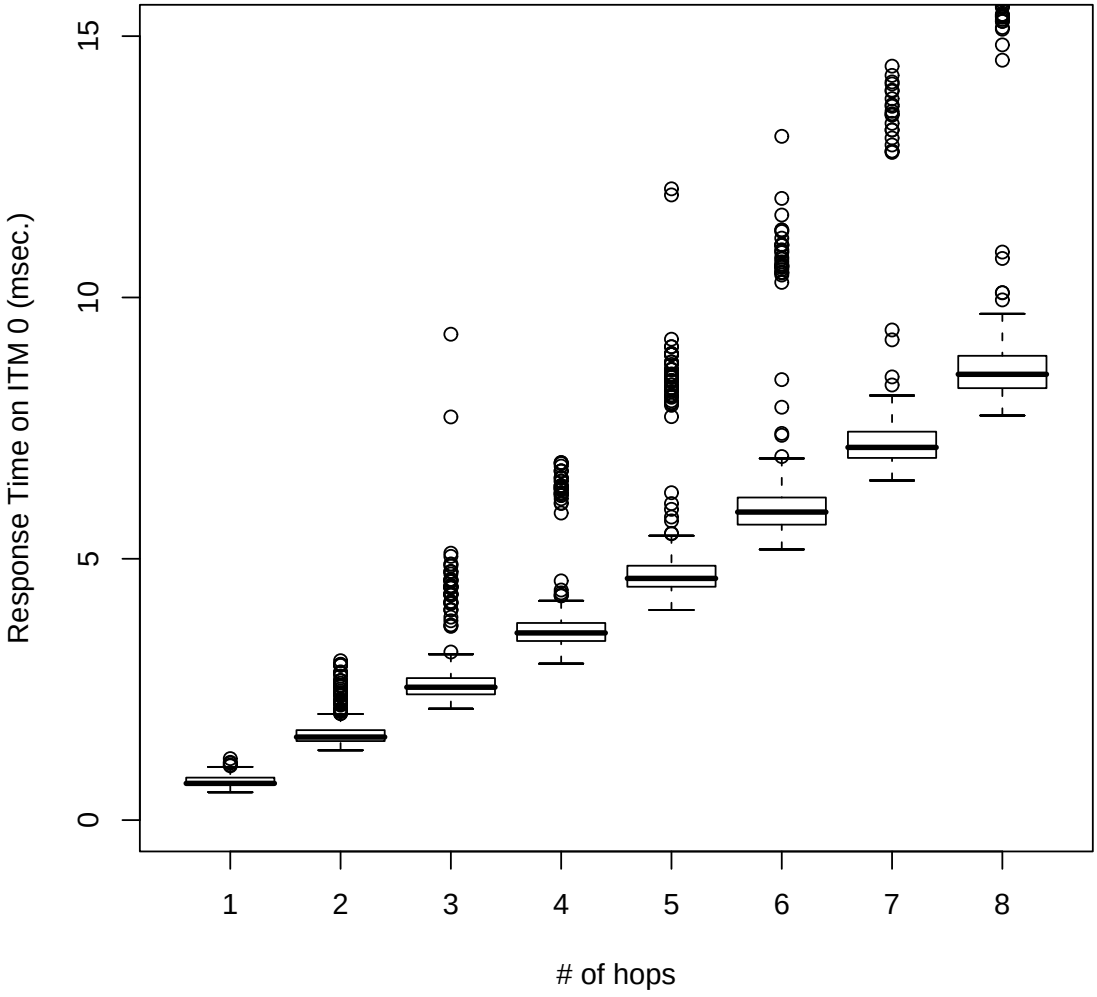


Figure 3.13: RTT of a ITM trace request (scope on the box)

90th percentile. The boxes in Fig. 3.13 show the curve of an increasing function.

Figs. 3.14 and 3.15 draw the distribution of the RTT on each ITM in the 9hop length topology. The tendency of the distribution was decreased along with the distance from the original issuer ITM as in the formula described in Section 3.8.1. Fig. 3.16 shows the distribution of the processing time on the border tracking stage in the 9hop length topology. The distribution expresses the time from receiving a request to forwarding the request to the upstream neighbor (i.e., $t_{dec(i)} + t_{btm(i)} + t_{req(i)}$). In this experiment, the tracking initiation stage on the original issuer ITM (ITM 0) was cut, that is, $t_{dec(i)} = 0$; therefore, the distribution on the ITM 0 was lower than other transit ITMs (ITM 1 to ITM 7). On the other hand, the ITM 8 was seen as the attacker AS. Because the attacker AS does not forward the request further upstream, the $t_{req(8)}$ is zero. For this reason, the distribution on the ITM 8 is lower than that of other transit ITMs. The processing time on the dummy BTM function was less than 400 microseconds on each ITM. Fig. 3.17 shows an example of the distribution of the dummy BTM function on the ITM 0. According Figs. 3.13 to Fig. 3.17, the formula described in Section 3.8.1 is adequate for expressing the response time of an traceback query.

In this evaluation, the RTT of an ITM trace request message with an actual implementation of hash-digest-logging method could not be evaluated because of the limited resource of the testbed environment. As sample data, we measured the response time of the software PAFFI [16], which was run on a VMware Workstation 5.0 [90]. We measured PAFFI's response time in 1,000 trials both in the case of *Found* and in the case of *Not Found* in the environment shown in Table 3.4. Table 3.5 shows the results.

According to the Table 3.5, the order of the response time of PAFFI is a thousand times as large as the RTT of our ITM implementation; therefore, the border tracking stage would become the bottleneck point on the trial of the inter-domain traceback. We estimated the RTT of an ITM trace request using the result of experiments and the formula described in Section 3.8.1. In the case where each AS used PAFFI as BTS in the 9hop length topology, the estimated RTT was about 16 seconds in average. Along with this preliminary evaluation result, it was concluded that the bottle neck point of the traceback trials through InterTrack would be the border tracking stage on each AS.

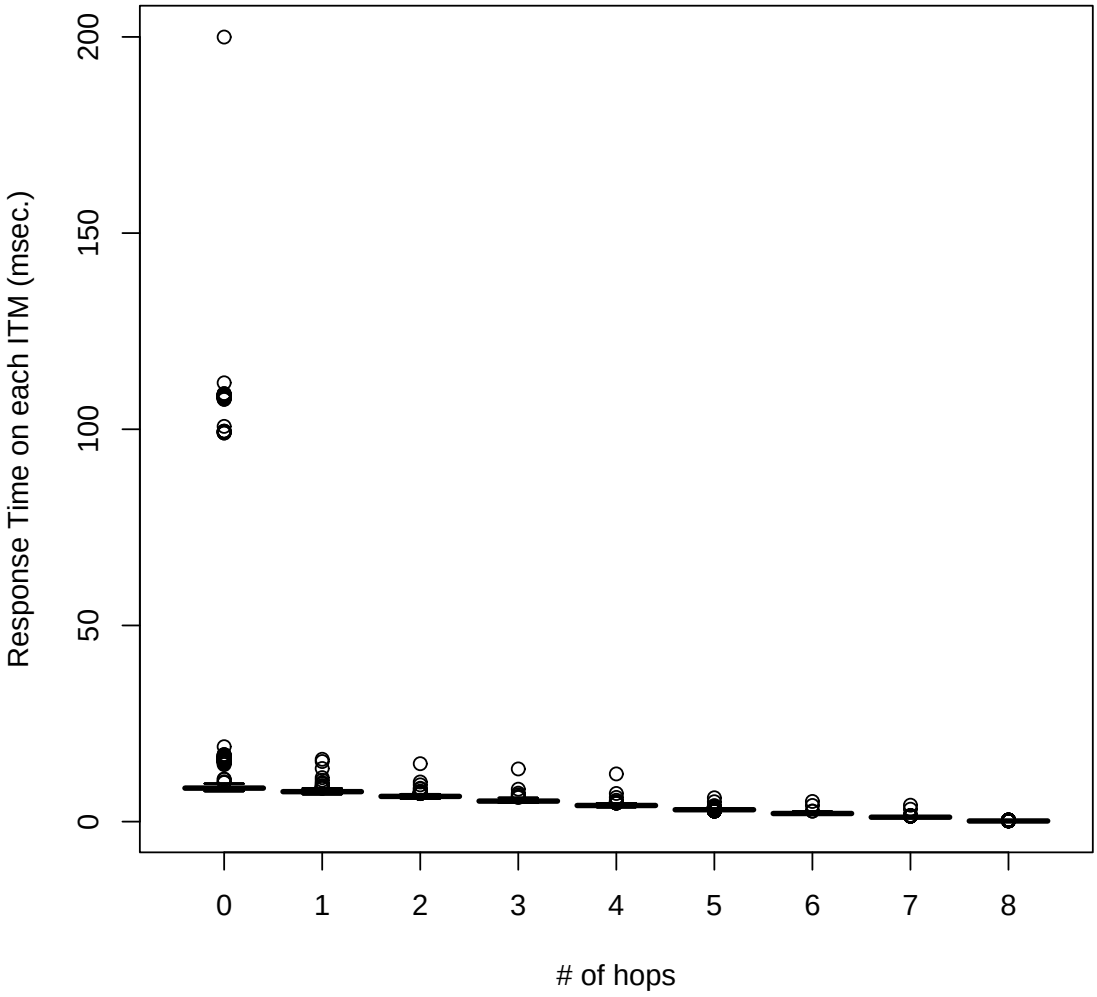


Figure 3.14: RTT on each ITM in a 9hops length topology

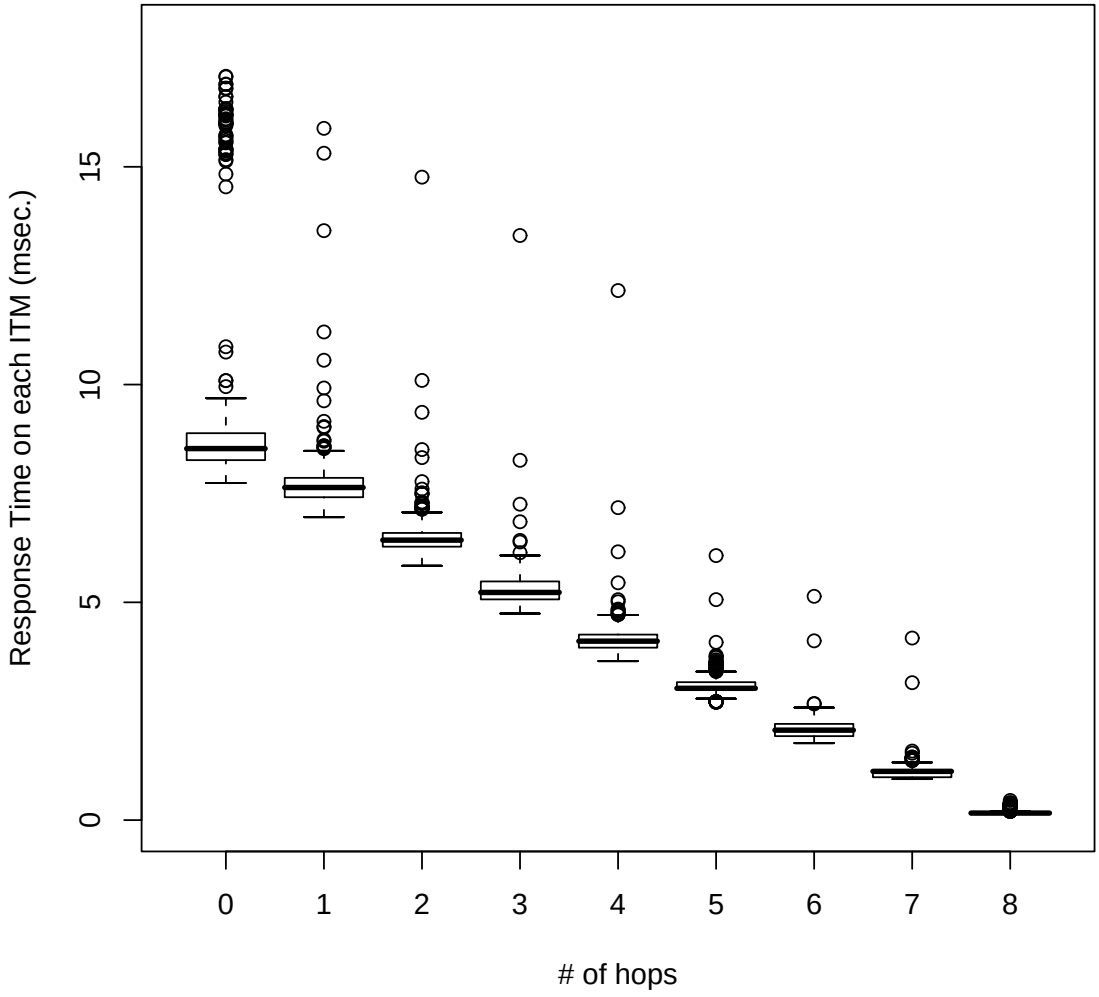


Figure 3.15: RTT on each ITM in a 9hops length topology (scope on the box)

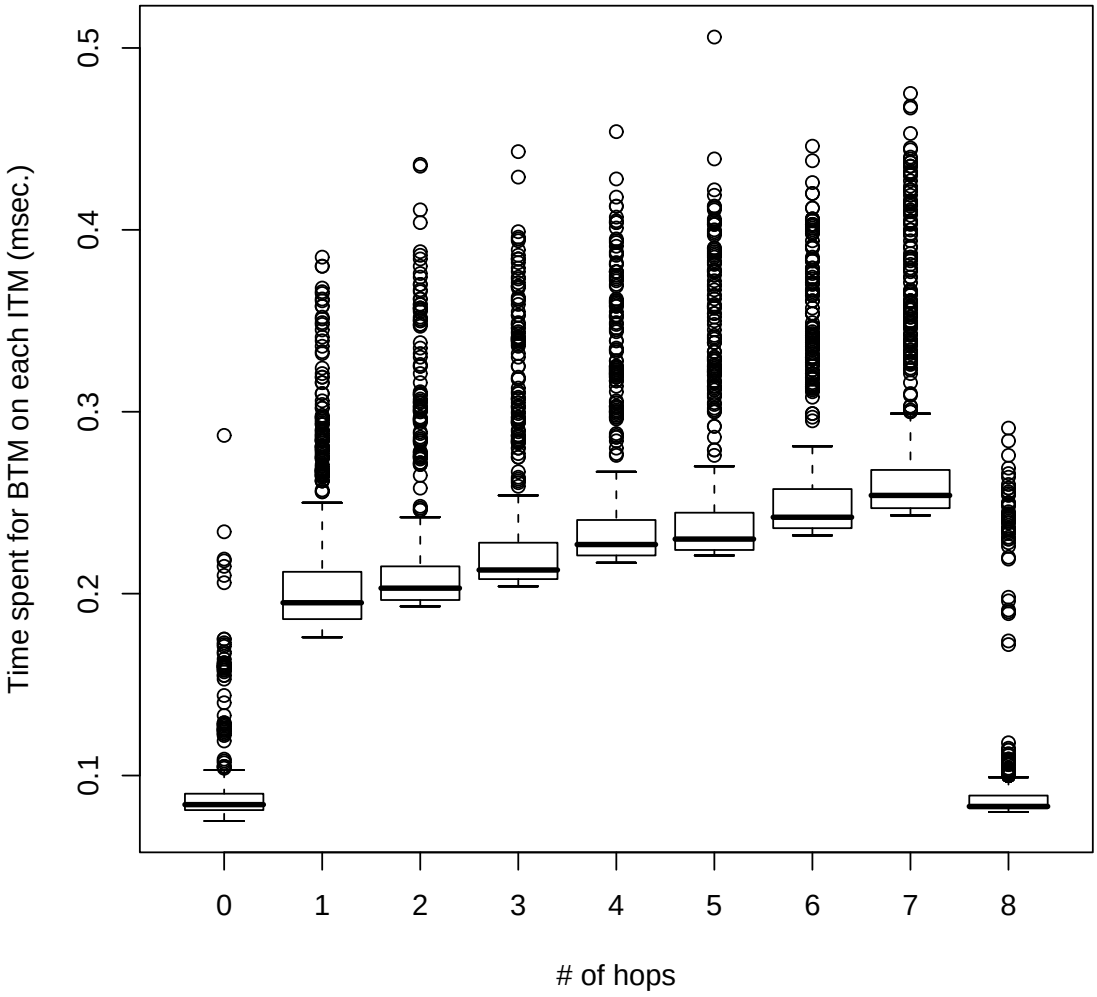


Figure 3.16: The processing time of the border tracking stage

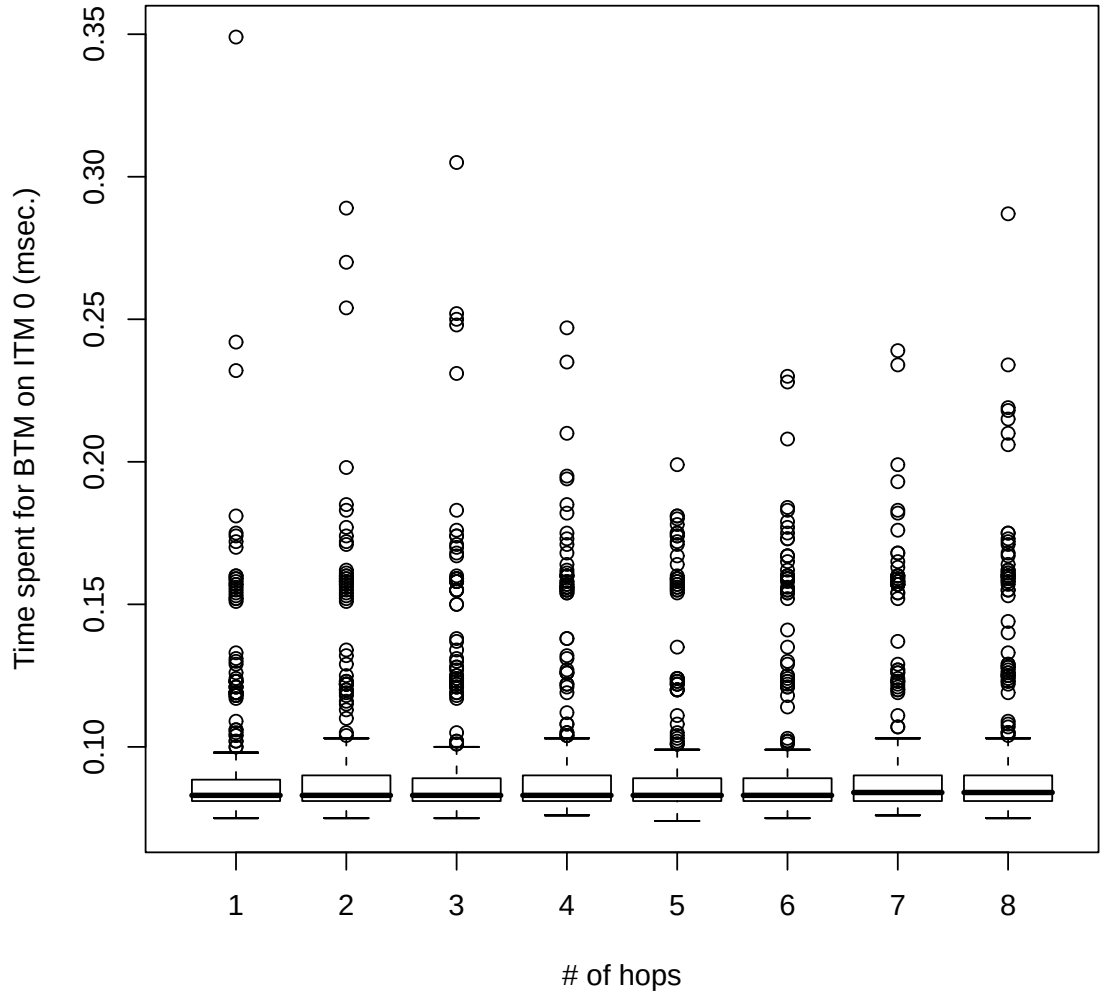


Figure 3.17: The processing time of dummy BTM function

Table 3.3: The data status of the box-whisker plots on Fig. 3.17

hop length	1	2	3	4	5	6	7	8
max. (msec.)	0.349	0.289	0.305	0.247	0.199	0.23	0.239	0.287
mean (msec.)	0.087	0.087	0.088	0.888	0.087	0.087	0.088	0.088
90th percentile (msec.)	0.0980	0.103	0.100	0.103	0.099	0.099	0.103	0.103
75th percentile (msec.)	0.0885	0.090	0.089	0.090	0.089	0.089	0.090	0.090
median (msec.)	0.0830	0.083	0.083	0.083	0.083	0.083	0.084	0.084
25th percentile (msec.)	0.0810	0.081	0.081	0.081	0.081	0.081	0.081	0.081
min. (msec.)	0.0750	0.075	0.075	0.076	0.074	0.075	0.076	0.075
variance	0.0002	0.0003	0.003	0.0002	0.0002	0.0002	0.0002	0.0003

Table 3.4: The spec of PAFFI on VMware

Item	Spec
Host PC	PowerEdge 1550
CPU	Pentium III 993 MHz
Host OS	Windows XP SP2
Guest OS	Red hat 6.2
Memory	768 MBytes
Network	VMNAT

Table 3.5: Response time of the software version PAFFI on VMware

	Found case	Not Found case
Max (sec.)	3.27	3.99
Min (sec.)	0.58	0.19
Mean (sec.)	1.57	1.61
Median (sec.)	1.58	1.59
Var	0.04	0.10

3.9 Comparison among Other Architectures

In this section, we compare InterTrack with other inter-domain traceback architectures. For comparison, we take the architectures which have been discussed in IETF: iTrace-CP [91] as one of the extensions of ICMP traceback message, SPIE [65], eIP/iIP [68], RID [67], RID-MEW [71] and InterTrack. Tables 3.6 and 3.7 show the comparison among each architecture in the qualitative view and Table 3.8 describes that of the quantitative view.

Except for iTrace, each architecture is a hierarchical structure. Each structure is characterized by the message protocols for the inter-domain traceback. On the reverse path of a traceback result, except for eIP/IP and InterTrack, each architecture reveals a topology or IP / MAC address to others. RID, RID-MEW and InterTrack are independent from a specific traceback technique.

iTrace-CP and eIP/iIP need the cooperation among ASes to collect traceback information for the reverse path reconstruction because of the characteristic of the traceback packet method. On other architectures, each AS can collect traceback information by itself only.

Authenticating and authorizing components, SPIE, eIP/iIP, and InterTrack are based on the certificate process of sBGP or soBGP and use IPSec ESP to encrypt a TCP session. RID and RID-MEW are authorized and authenticated by the CAs operated by consortium and employ SOAP or HTTPS for the message encryption. We suppose that InterTrack is comparable or may be superior with others in each item of the Tables 3.6 and 3.7.

On quantitative view (Fig. 3.8), several architectures have reported the expected

Table 3.6: A comparative table on the qualitative view (1)

approach	protocol	structure	purpose	reverse path	disclosed information
iTrace-CP	ICMP traceback message	flat	layer 3 traceback of traffic	router hops	IP, MAC, topology
SPIE	IP Packet Traceback Protocol	single level hierarchy	layer 3 traceback of a packet	agent (router) hops	IP, topology
eIP/iIP	ITM Protocol	two level hierarchy	layer 3 traceback of traffic	AS hops (eIP) router hops (iIP)	AS number, AS path
RID	IODEF RID extension	consortium peering	layer 3 traceback, Incident report	AS hops	AS number, AS path, IP
RID-MEW	RID MEW extension	three level hierarchy	layer 3 traceback, incident report	AS hops	AS number, AS path, IP
InterTrack	ITM trace messages	two level hierarchy	layer 2/3/7 traceback	AS hops	AS number, AS path

Table 3.7: A comparative table on the qualitative view (2)

approach	dependent technique	cooperation with others to collect evidences	traceability of messages	authentication	based certification
iTrace-CP	traceback packet	needed	router ID	—	—-
SPIE	hash-digest-logging	not needed	message ID, signatures	IPSec	sBGP
eIP/iIP	traceback packet	needed	HMAC, signatures	HMAC, IPSec	sBGP, soBGP
RID	independent	not needed	message ID, signatures	SOAP/HTTPS	consortium
RID-MEW	independent	——-	message ID, signatures	SOAP/HTTPS	—
InterTrack	independent	not needed	message ID, ITM path list	IPSec	sBGP, soBGP

Table 3.8: A comparative table on the quantitative view

	iTrace-CP	eIP/iIP	RID-MEW	InterTrack
experiment	ns2	emulation with implementation	emulation with dummy function	emulation with dummy function
average time spent to trace	18.3 sec.	38 sec.	1.6 sec. (test)	0.25 sec. (test) 16 sec. (estimated)
hop length	20 (router)	4 (AS)	7 (AS)	9 (AS)
trees	single tree	9 trees	single tree	single tree
CPU	— —	Pentium III 800 MHz	Pentium IV 3.0 GHz	Pentium III 1.4 GHz
tracing	ns2	actual implementation	dummy (fixed value)	dummy (flexible value)
tracing time on each AS	—-	—	0.2 sec. (mid. hop) 0.4 sec. (end hop)	0.083 usec. (test) 1.6 sec. (PAFFI)

time to finish a traceback trial. Although each experimental environment is different from each other, the average time to spend on a traceback trial was less 40 seconds on iTrace-CP, eIP/iIP, RID-MEW and InterTrack. Even when the sample data on the response time of PAFFI is considered, the average time of a traceback trial on InterTrack is comparable to others.

Through these comparisons, InterTrack can be considered a competitive traceback architecture in practical use.

3.10 Summary

Developing an automated inter-domain traceback architecture has long been viewed as impractical due to the barriers on the operational boundaries and the dependence on specific tracking techniques. InterTrack's key contribution is to achieve the inter-domain traceback in a practical way. Because of the phased-tracking apporach of InterTrack, each AS can control a traceback operation on its operation domain by itself only, and can track back an attack on its operation domain with different traceback techniques regardless of traceback systems on other ASes. The federation of internal traceback trials on InterTrack enables ASes to cooperate with others on an inter-domain traceback trial automatically, while concealing the sensitive information of each AS.

Through preliminary experiments, the time spent for an inter-domain traceback in 9 AS hop length was estimated to be 16 seconds in the case where each AS uses hash-digest-logging method. Through discussions and comparisons among other traceback architectures, we explained InterTrack as the competitive traceback architecture against attacks.

In future work, in order to confirm the feasibility of InterTrack, more complex cases where ASes employ various traceback techniques such as specialized routing methods [11, 25, 42, 43], flow sampling methods [12, 13], hash-digest-logging methods [16, 29] etc., will be evaluated.

Chapter 4

A Binding Technique for the Intra-Domain Traceback Techniques

4.1 Introduction

In this chapter, we explore intra-domain traceback techniques for the intra-AS tracking stage, that is, for a deep internal investigation to locate attacker nodes.

Unfortunately, the anonymous nature of the IP protocol makes it difficult to accurately identify the IP address of an attacker node, if the attacker wishes to conceal it. Essentially, the amount of the traffic of a DDoS attack around an attacker node is smaller than that around a victim node, therefore, intra-domain traceback techniques should have the capability to detect an attacker node from a small amount of traffic generated by attacking flows.

According to the Section 2.1.2, hash-digest-logging method is able to trace a single packet. In this technique, each router audits all packets forwarded, and stores *packet digests* instead of the packets themselves. Packet digests are stored in *digest tables*, which are bitmaps based on a space- and time-efficient data structure known as a Bloom filter [40]. Digest tables reduce the required storage size by several orders of magnitude over current log-based techniques [24]. To trace an attack packet, the attack path is reconstructed by checking digest tables on each router. The hash-digest-logging method is suitable for identifying an attacker on intra-domain networks because of its

ability to trace a single packet.

An attack path determined by a hash-based IP traceback indicates the ingress point of an attack packet. However, the ingress point is the nearest router of the attacker node on a network, not the attacker node itself, because of limitations of the algorithm used to store packet digests. The technique thus cannot identify the attacker node itself on the subnet.

On the other hand, the FDB-checking method is a useful technique to locate attacker nodes on a layer 2 network. The FDB-checking method can identify the location of an attacker node by the source MAC address of an Ethernet frame. Because of the characteristics of the learning manner of the FDB, the FDB-checking method can track an Ethernet frame even when the source MAC address of the Ethernet frame was spoofed. Instead of these characteristics, FDB-checking is effective only when the targeted MAC address was learned on every layer 2 switch on a layer 2 network. Furthermore, FDB has an aging timer to avoid memory consumption by unused MAC addresses. If the targeted MAC address was aged out, the FDB-checking method cannot locate the attacker node. Furthermore, an attacker node puts the blame on another node by hijacking the IP address and the MAC address assigned to the other node on the same layer 2 network.

In this section, we propose an ingress-port-based layer 2 traceback technique. Our layer 2 traceback technique, called here the *ingress-port-based tracking (IPBT)* method, is based on the hash-digest-logging technique of the Source Path Isolation Engine (SPIE) proposed by Snoeren et al. [29]. The IPBT method records a packet digest with the identifier of the ingress port (or the ingress interface) of the packet on a layer 2 switch or a leaf router. IPBT records packet digests and the corresponding port identifiers into digest tables. Instead of the source MAC address of an Ethernet frame (or a packet), the IPBT method is not influenced by the aging out of the FDB on each layer 2 switch. The identifier of each port is fixed value, therefore, attackers hardly spoof it.

We also propose a layer-2 extension to SPIE (L2-SPIE) as an interconnecting method between the hash-digest-logging method and the IPBT method. To narrow down the existence range of an attacker node, and eventually to detect the attacker node itself, the attack packet has to be traced on the layer-2 network under the ingress point router. In our extension, SPIE-enhanced leaf routers on the intra-domain network store packet digests by the IPBT method for the traceback on layer 2 networks

while storing packet digests by the hash-digest-logging method for the traceback on layer 3 networks. If an attack packet passes through a layer 2 switch, the recorded layer 2 information distinguishes the ingress port of the packet on the switch.

Of course, holding layer 2 information with one-to-one correspondence to a packet digest requires a large amount of storage space. To reduce storage requirements, two different digest tables are provided. One is an *L3 digest table*, which stores packet digests themselves. The other is an *L2 digest table* which records layer 2 information corresponding to packet digests. These tables permit cooperation between the layer 3 traceback and the layer 2 traceback, and enable a more detailed tracing of attacker nodes.

4.2 Source Path Isolation Engine (SPIE)

Snoeren et al. have proposed hash-based IP traceback and Source Path Isolation Engine (SPIE) architecture [29], which is an implementation of the hash-digest-logging method. SPIE uses auditing techniques to support the traceback of individual packets, while reducing the storage requirements by several orders of magnitude over current log-based techniques [24].

Fig. 4.1 shows the architecture of SPIE. SPIE has a hierarchical architecture. STM (SPIE Traceback Manager) controls the entire SPIE system. The STM is the interface to the intrusion detection system or other entities requesting a packet trace. When a request is presented to the STM, it verifies the authenticity of the request, dispatches the request to the appropriate SCARs (SPIE Collection And Reduction agents), gathers the resulting attack graphs, and assembles them into a complete attack graph. Upon completion of the traceback process, the STM replies to the intrusion detection system with the final attack graph.

SCARs are responsible for a particular region of the network, serving as data concentration points for several routers and facilitating a traceback of any packets that traverse the region. Due to the complex topologies of todayFs ISPs, there will typically be several SCARs distributed over an entire network. Upon request, each SCAR produces an attack graph for its particular region. The attack graphs from each SCAR are grafted together to form a complete attack graph by STM.

Each SPIE-enhanced router has a Data Generation Agent (DGA) associated with it. Depending upon the type of router in question, the DGA can be implemented and

deployed as a software agent, an interface card plug to the switching background bus, or a separate auxiliary box connected to the router through some auxiliary interface. The DGA produces packet digests of each packet as it departs the router, and stores the digests in time-stamped digest tables. The tables are paged every so often, and represent the set of traffic forwarded by the router for a particular interval of time. Each table is annotated with the time interval and the set of hash functions used to compute the packet digests over that interval. The digest tables are stored locally at the DGA for some period of time, depending on the resource constraints of the router.

4.2.1 The Characteristics of the Auditing Techniques on SPIE

In SPIE architecture, a Data Generation Agent (DGA) on each router stores packet digests, which are n-bit hash values, instead of the packets themselves.

An input value for hash functions to make packet digests, here called the *packet signature*, is a masked IP header plus the first 8 bytes of payload (Fig. 4.2). A packet signature uniquely represents an IP packet and enables the identification of the packet across hops in the forwarding path.

To reduce storage requirements, the DGA stores sets of packet digests in a space-efficient data structure known as a Bloom filter [40]. The Bloom filter computes k distinct packet digests for each packet, using independent uniform hash functions, and stores the n-bit results as bit positions in a digest table, a 2^n-sized bit array constructed by the Bloom filter. The bit array of a digest table is initialized to all zeros, and bits are set to one as packets are received. Figure 4.3 shows a Bloom filter with k functions. A digest table is dispatched when it is full or when a certain time interval for refreshing hash functions (refreshing time) is reached. The dispatched digest table is stored in a ring buffer on the router, with a time stamp and a set of used hash functions. To avoid hash collisions, each router selects a new set of k distinct hash functions at the refreshing time.

The traceback process starts at the 1-hop upper router of a victim node. First, the DGA on a SPIE-enhanced router computes the k hash digests on the packet in question and checks the indicated bit positions of the appropriate digest table. If any bit position is zero, the packet was not forwarded by the router. However, if all the bits are one, it is highly likely that the packet passed through the router. When it appears that the router forwarded the suspected packet, neighbor routers' digest tables are

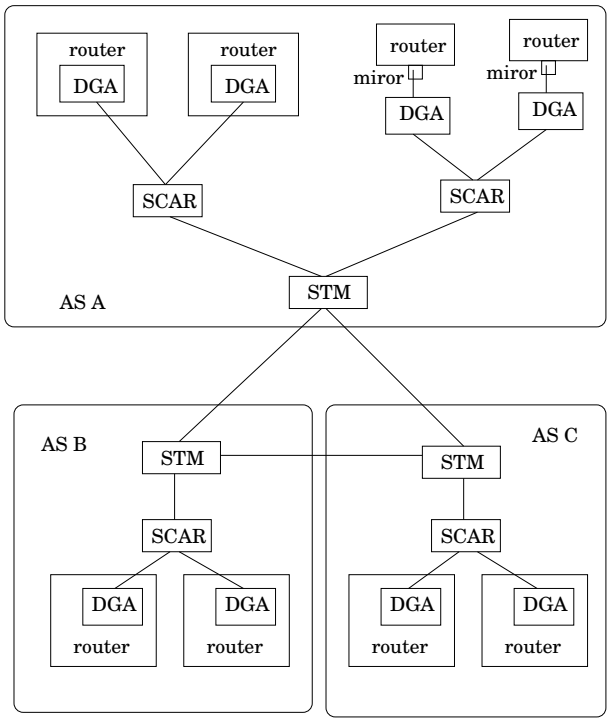


Figure 4.1: SPIE Architecture

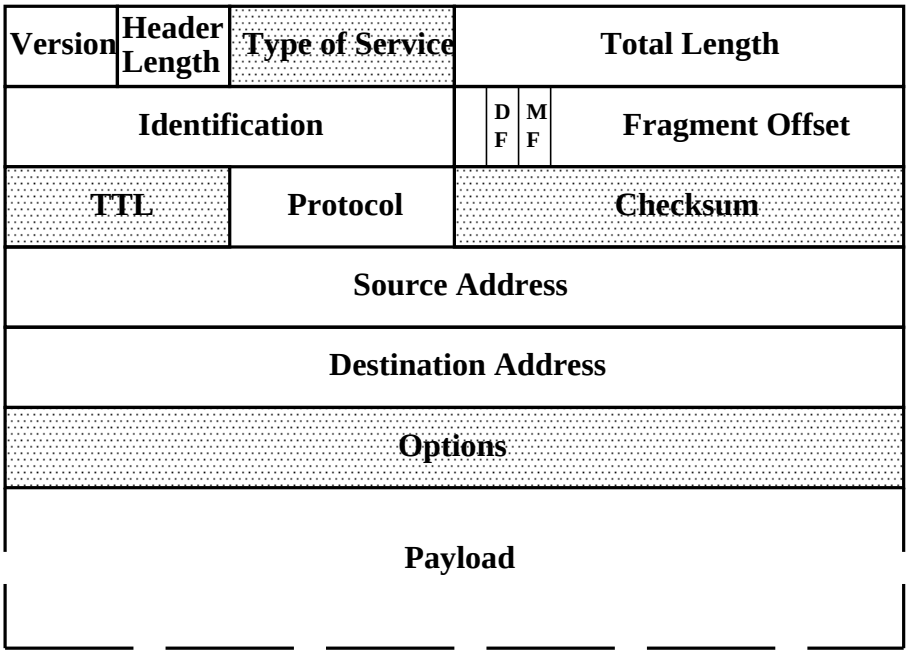


Figure 4.2: Packet Signature

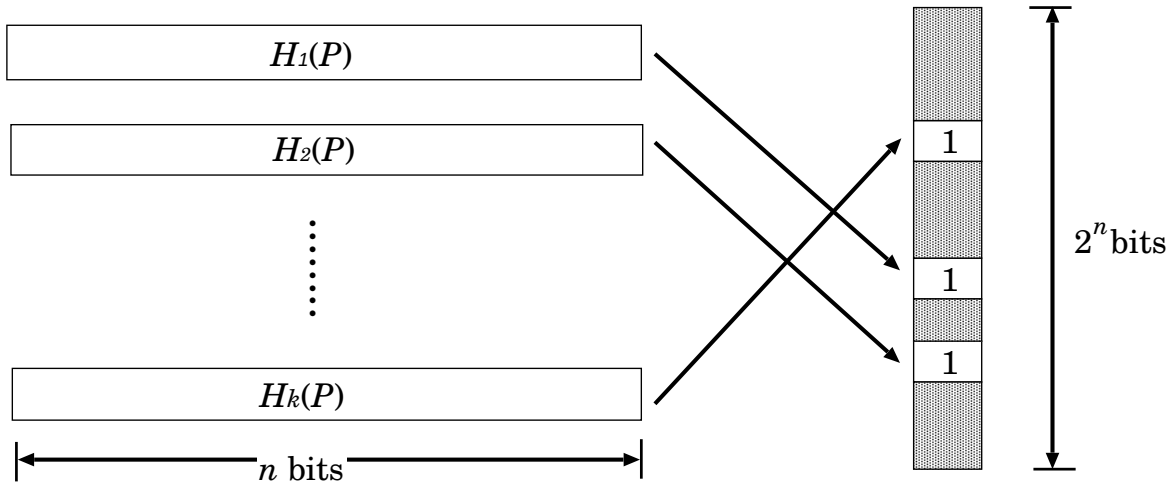


Figure 4.3: Bloom Filter

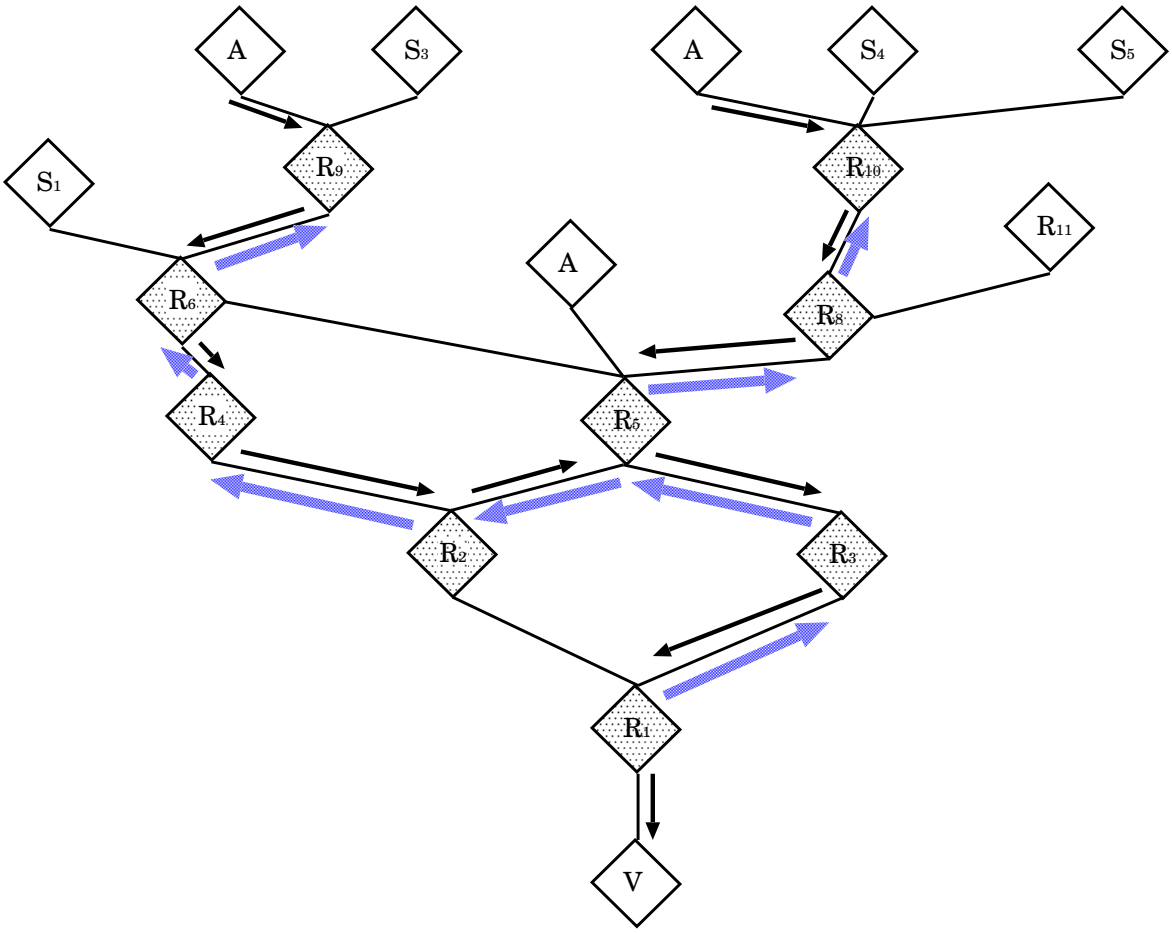


Figure 4.4: SPIE traceback process

then tested. This process continues until all branches of the attack graph are checked. Figure 4.4 shows an example of this traceback processing. Tracing proceeds from R_1 (the victim's neighbor router) towards the attacker node, A. Black arrows represent the path of the attack packet, and gray arrows represent the reconstructed attack graph, respectively. The ingress point of the attack is R_{10}, this is the nearest router to the attacker A. When an attacker node is in the SPIE-enabled network, the ingress point of an attack packet is a leaf router within the network. Otherwise, a border router of the SPIE-enabled network is the ingress point of the attack.

The false positive rate of a digest is given by several parameters. When a Bloom filter takes m bits of memory, which can store n packets, the effective false-positive

rate with k digest functions can be expressed as

$$P = (1 - (1 - \frac{1}{m})^{kn})^k \approx (1 - e^{-kn/m})^k$$

Tables are available which provide the effective false-positive rates for various capacities and the number of digesting functions [92]. The size of a Bloom filter is defined by a capacity factor $c(= m/n)$; in other words, storing n packets requires a Bloom filter of the size $m = cn$. If a 2M byte ($= 2^{24}$ bit)-sized digest table is taken with a capacity factor of five), it can store roughly 3.2M packets. Considering the trade-off between the false positive rate and the storage requirements, Snoeren et al. [29] have proposed using a Bloom filter with three digesting functions ($k = 3$) and a capacity factor of five ($c = 5$) in practice. In this case, assuming the average size of a packet is approximately 1000 bits, SPIE requires roughly 0.5% of the total link capacity in digest table storage with a rate of 0.092 false positives.

Also, Snoeren et al. mentioned the access rates on the memory to read and write digest tables. A router processing more than 20Mpkts/sec require an SRAM digest table because of the limited DRAM cycle time. Furthermore, higher speed routers must page digest tables to SDRAM in order to store a minute's worth of digests. In some border cases on slower routers with many neighbors, Snoeren et al. has reported that it may be more practical to use a larger amount of slower memory and reduce the number of memory accesses required per packet, allowing DRAM to be used instead of SRAM, for example. If a hash table of b-bit values is used to record n packet digests using $1.44n$ b-bit entries, the required memory size is $m' = 1.44nb$ bits. Such a table is less than 70% full; hence, each packet insertion takes only $\sim$2 memory accesses in expectation. The memory required for a particular false-positive rate P while storing n packets is given by $m' = 1.44n \cdot log(n/P)$ and the additional cost of using a hash table instead of a Bloom filter, in terms of increased memory consumption, is a factor of (for small values of P) $m'/m = 1 + log_{1/p}(n)$.

4.2.2 Limitations of SPIE

The ingress point detected by the SPIE is either a border router of the SPIE system or the leaf router, which has the attacker node on its subnet, however, the SPIE can not locate the attacker node itself. Examining digest tables of a leaf router never reveals which node on a subnet is the attacker node, because a digest table shows only whether or not the router forwarded the packet in question.

Specifying only an ingress point leaf router is not sufficient to stop a DDoS attack. Filtering, based on the destination (victim's) IP address, TCP ports,etc., blocks not only the attack packets but also the packets sent from other (normal) nodes; hence, such filtering denies other nodes' communications or services. Leaving attacker nodes on subnets permits later DDoS attacks. To terminate current attacking flows from a network without blocking other nodes' communications, and to prevent other DDoS attacks in the future, attacker nodes themselves must be discovered and physically removed from the network.

Tracing attack packets on a layer 2 network, however, is difficult. If the subnet under the leaf router is widely spread, as in an xDSL network or in a Metro-ether network, human resources and a lot of time are required to identify the attacker node and remove it from the subnet network. Therefore, to identify the attacker node itself, some method is required to trace the attacker node on the subnet L2 network under the leaf router.

Theoretically, if hash checkers like DGA are set against all nodes on a subnet, they can identify the true attacker node. However, from an operational point of view, this is not a realistic approach.

4.3 Limitations of FDB-Checking Method

Each Network Interface Card (NIC) has a globally unique address, the Media Access Control (MAC) address, in which the upper 24 bits comprise the vendor code and the lower 24 bits comprise an address assigned by the vendor. IEEE, the official global authority, is responsible for handing out blocks of MAC addresses. On Ethernet networks, which constitute today's local area networks, a packet is forwarded on the basis of MAC addresses.

The source MAC address of an incoming packet is available for tracing the attacker node on the subnet of a leaf router. The source MAC address of an incoming packet is the MAC address assigned to the device of the sender node. Moreover, a switch maintains a database of all MAC addresses received on all of its ports, and uses database entries to decide whether a packet frame should be forwarded or filtered. This database is called an L2 forwarding database (FDB).

Each FDB entry consists of the MAC address of the device, an identifier for the port on which it was received, and an identifier for the VLAN to which the device belongs.

The switch can learn FDB entries; that is, the system of the switch updates its FDB with the source MAC address from a packet, the VLAN, and the port identifier on which the source packet is received. Looking up the FDB on a switch tells one the port on which a source MAC address has been learned. Whether or not a source MAC address is spoofed, the FDB temporarily records the source MAC address paired with the incoming port. By searching the port which learned the source MAC address of an attack packet, and tracing switches in a hop-by-hop fashion, one can locate the attacker node or one can detect the port to which the attacker node directly connects. FDB-checking method [18–23, 37, 38] uses these characteristics of FDB.

By combining the hash-digest-logging method and the FDB-checking method, there is the possibility of tracing an attack packet to the port of a layer 2 switch to which the attacker node connects; however, large amounts of storage on the leaf router are required to store source MAC addresses which correspond to packet digests. Furthermore, the FDB cannot learn more than a certain number of MAC addresses, because the available memory size for the FDB is limited. To prevent the FDB from becoming full, the FDB has an *aging time*. Entries in the FDB are removed (aged out) if, after a period of time (the aging time), the device has not transmitted. When an attacker spoofs the source MAC address of an attack packet, and the spoofed MAC address has been "aged out" from the FDB, there is no way to trace the attacker node by using stored MAC addresses.

4.4 An Ingress-Port-Based Tracking Method for a Layer 2 Traceback

In this section, we propose an *ingress-port-based tracking* (IPBT) method for a single packet traceback on a layer 2 network. The IPBT method is based on the hash-digest-logging techniques of SPIE. Extending the Bloom filter techniques of SPIE, IPBT stores and looks up the identifier of the ingress port which received an issued attack packet.

The IPBT method runs on a layer 2 switch or a leaf router. The IPBT method records packet signatures with FDB information into digest tables. The sequence of recording mechanisms of the IPBT method is as follows:

(1) A layer 2 switch receives an Ethernet frame.

(2) The layer 2 switch stores the Ethernet frame on a temporary buffer while forwarding the Ethernet frame to the destination.

(3) The layer 2 switch looks up the identifier of the ingress port which learns the MAC address of the Ethernet frame.

(4) The layer 2 switch makes the packet signature from the Ethernet frame stored on the temporary buffer.

(5) The layer 2 switch concatenates the ingress port identifier and the packet signature, and records them into digest tables.

On the other hand, the tracking mechanism is as follows:

(1) The agent on a layer 2 switch receives a query with a packet signature of the issued attack packet.

(2) The agent combines the packet signature and the identifier of each port and verifies the records on digest tables. The agent checks whether each combination is recorded on digest tables or not.

(3) The agent replies to all port identifiers which hit on digest tables.

On making a packet signature, IPBT doesn't use Ethernet header. Generally, an Ethernet header is useful on the same layer 2 network. When a packet is forwarded to different layer 2 network, a router changes the Ethernet frame to forward the packet to the next hop router or an end node. Hence, nobody on the victim's site can get the Ethernet header of an attack packet which is the same Ethernet frame initially used by the attacker node. Because of the aging timer of FDB, recording whole Ethernet header corresponded to each packet consumes a large amount of memory or storage. Thus, we exclude the Ethernet header from the input values of hash functions.

Instead of Ethernet headers, IPBT uses port identifiers with packet signatures as hash inputs. There are trade-offs between the steps on looking up port identifiers from digest tables and required memory size. If IPBT uses a hash table of b'-bit to store the pair of a packet digest and a port identifier, the size of the hash table recording n packets with P false positive rate is given by $m'' = 1 + log_{1/P}(n) + b'$ bits. The digest table will be used only when an incident occurs; therefore, digest table wastes the memory of a layer 2 switch if no incident happens in a time interval. Hence, the required memory size for digest tables on IPBT should be as small as the required memory size on hash-digest-logging method, although IPBT algorithm takes $O(n)$ to find the ingress port's identifier from a digest table.

In addition to the memory consumption, the IPBT algorithm can avoid hash collisions. If attacker nodes on different ports send attack packets which generate the same packet signature in order to cause hash collisions, the IPBT distinguishes port identifiers of each attacker node because each bit position indicated by the packet signature and each port identifier will be different due to the characteristics of one way hash functions.

4.5 A Layer-2 Extension of SPIE

Next, we present a layer-2 extension of SPIE (L2-SPIE), which narrows down the existence range of an attacker node on a subnet layer 2 network, while reducing other storage requirements for the extension. In this section, we explain the interconnecting technique of L2-SPIE between the hash-digest-logging method of SPIE and the IPBT method. L2-SPIE binds a layer 3 traceback on SPIE and a layer 2 traceback on IPBT by two convert tables and two types of Bloom filters.

4.5.1 Assumptions

Before describing the details of L2-SPIE, several assumptions are defined:

- SPIE has detected an ingress point leaf router on the intra-domain network;
- MAC addresses are used to resolve hosts on subnets under the leaf router;
- All switches along the path from the leaf router to the attacker node support Bridge-MIB [93];
- The source IP address and the source MAC address of a packet may be spoofed.

The first assumption is made because SPIE is used to identify the subnet layer 2network on an intra-domain network where the issued attacker node exists. The second and third assumptions reflect the fact that L2-SPIE is based on MAC addresses and FDBs of layer 2 switches, and that L2-SPIE uses Bridge-MIB to obtain FDBs from switches. Bridge-MIB is widely supported by today's layer 2 switches. In the fourth assumption, the ability of an attacker is anticipated.

4.5.2 Identifiers

L2-SPIE uses two identifiers instead of storing MAC addresses. Using two conversion tables, L2-SPIE detects the port of a switch through which the attack packet entered. The two identifiers are as follows:

- NI-ID ($8bit$): a network identifier assigned to either the MAC address of a network interface on a leaf router, or the *VLAN ID* of a virtual interface if the leaf router uses VLAN interfaces.

- Port-ID ($12bit$): a port identifier assigned to a port of a switch. Each port has a different Port-ID.

A network interface identifier (NI-ID) is assigned to each MAC address of the network interface card. The NI-ID specifies which subnet an attack packet comes from. Currently, a router can equip 802.1Q VLAN interfaces, which means that multiple virtual interfaces can be used on one physical network interface. Although VLAN interfaces, which share one physical interface, have the same MAC address, each VLAN interface has a different *VLAN ID*, so that a router distinguishes VLAN interfaces by their *VLAN IDs*. If a network interface is a VLAN interface, an NI-ID is assigned to its *VLAN ID* instead of the MAC address shared with another VLAN interface.

A port identifier (Port-ID) is assigned to a port on a particular slot of a layer 2 switch through which a leaf router directly connects. A Port-ID tells one through which port of a layer 2 switch an attack packet came in.

A combination of an NI-ID and a Port-ID distinguishes a particular port from other ports if several network interfaces of a leaf router connect to the same layer 2 switch.

An 8-bit NI-ID, expressed in the range from 0 to 255, is sufficient to represent the network interfaces of a leaf router. A 12-bit Port-ID, whose range is 0 to 4095, is sufficient to express all ports of a current switch.

4.5.3 Conversion Tables

L2-SPIE uses two conversion tables to look up NI-ID and Port-ID from a MAC address: the NI-ID table, and the Port-MAC table.

A leaf router keeps an NI-ID table that is based on the interface configuration. Each NI-ID table entry consists of an NI-ID and the MAC address of a network interface card assigned with the NI-ID. If VLAN interfaces are used, then the entry in the NI-ID table consists of an NI-ID and a *VLAN ID*. Table 4.1 shows the NI-ID table of a leaf router which uses VLAN interfaces on a physical interface whose MAC address is "C".

A Port-MAC table, constructed by the FDB of a layer 2 switch that connects to the leaf router directly, is used to obtain the Port-ID from the source MAC address of a packet which enters the leaf router via the layer 2 switch (Table. 4.2). The leaf router obtains the contents of the FDB via Bridge-MIB [93]. Bridge-MIB is widely supported by current switches. An entry in the Port-MAC table is composed of a

Table 4.1: NI-ID table

NI-ID	MAC address	VLAN ID
1	A	
2	B	
3	C	101
4	C	102

Table 4.2: Port-MAC table

NI-ID	Port-ID	MAC address
1	2	F
	3	E
	4	H
	4	K

source MAC address and a Port-ID. The source MAC address is that of the incoming packet, and the Port-ID identifies the layer 2 switch's port that has learned the source MAC address. Recording Port-IDs instead of source MAC addresses, and specifying the incoming port of layer 2 switches by recorded Port-IDs, enables us to trace an attack packet even when the source MAC address has aged out from the FDB of a layer 2 switch.

Port-MAC tables are generated for each NI-ID. This indicates that an NI-ID and Port-ID pair identifies the port in which the attack packet came, even if several network interfaces of a leaf router connect to the same layer 2 switch. Table 4.2 shows the Port-MAC table paired with NI-ID "1". Two entries with the Port-ID "4" represent two source MAC addresses ("H" and"K") which have been learned on the layer 2 switch port whose Port-ID is "4".

4.5.4 Algorithms of the L2-SPIE

For each incoming packet, extended DGA (xDGA) on the leaf router searches identifiers from conversion tables corresponding to an incoming packet. To reduce memory

requirements for the storage of detected identifiers corresponding to the packet, xDGA uses an L2 digest table, another Bloom filter for storing identifiers corresponding to the packet. To use L2 digest tables, we provide a storing algorithm and a detecting algorithm for Bloom filters and for L2 digest tables. The former is used to encode identifiers into L2 digest tables, and the latter is used to decode identifiers from L2 digest tables.

Figure 4.5 shows an algorithm to check identifiers against each incoming packet and to store detected identifiers in an L2 digest table.

For an incoming packet, xDGA extracts source/destination MAC addresses from the Ethernet frame, and pulls up the packet signature from the packet. xDGA runs a process of storing packet signatures into L3 digest tables, while running a process to store identifiers in L2 digest tables.

In the process to store identifiers, xDGA first refers to the NI-ID table to search for an entry whose MAC address is the same value as the extracted destination MAC address of the packet, and to get the NI-ID assigned to the detected MAC address entry. Next, xDGA takes a PortMAC table corresponding to the NI-ID, and checks entries which have the same MAC address as the extracted source MAC address. If an entry is detected, the Port-ID of the entry represents the incoming port of the packet on a layer 2 switch. If not, xDGA sets the Port-ID to the value of NOT_FOUND_PORTID.

xDGA makes a hash input by concatenating the packet signature and detected identifiers, and computes hash values using hash functions for the Bloom filter of an L2 digest table. The bit positions on the L2 digest table, which are indicated by those hash values, are set to 1. This ensures that the pair of identifiers is stored in the L2 digest table corresponding to the packet.

The algorithm to derive two identifiers from L2 digest tables in the traceback process is shown in Fig. 4.6.

When an xDGA leaf router receives a suspected packet from the traceback manager STM, xDGA first generates the packet signature from the suspected packet. xDGA runs the IP traceback process and L2 traceback process (process to detect identifiers) in parallel. In the L2 traceback process, xDGA pulls up L2 digest tables whose time stamps are considered as the time when the suspected packet might have come through the leaf router, and checks each L2 digest table to establish whether it has a record of identifiers with the suspected packet's signature. The test of an L2 digest table is as follows: xDGA chooses an NI-ID and Port-ID pair, and generates a hash input to test

```
[ recording identifiers in L2DT procedure ]


receive an incoming frames

extract the source MAC address M_s
     and the Destination MAC address M_d from a frame
extract the packet signature PS from an packet
refer to NI-ID table
for (number of entries on the NI-ID table){
  extract an entry NE_i from the NI-ID table
  compare NE_i.ether_addr to M_d
  if (NE_i ≡ M_d){
    N_id = NE_i.id
    break the loop
  }
}
refer to a PortMAC table assigned to the Nid
for (number of entries on the PortMAC table){
  extract an entry PE_i from the PortMAC table
  compare PE_i.ether_addr to M_s
  if (PE_i ≡ M_s){
    P_id = PE_i.id
    break the loop
  }
}
make input values I by concatenating PS, N_id, and P_id
compute hash values H(I)_i  (1 ≤ i ≤ k)
for (number of hash values){
  search the bit indicated by H(I)_i on L2DT
  if (the bit ≡ 0){
    set the bit to 1 (that is, store identifiers on L2DT)
  } else {
    hash collision occurs, increment counter of duplication
  }
}
```

Figure 4.5: Procedure to store identifiers in L2DT

the packet signature of the suspected packet and the pair of identifiers. Then, xDGA computes hash values of each hash function, and examines each bit position pointed by hash values on the L2 digest table. If all the bits are 1, the ingress interface on the leaf router and the ingress port on a layer 2 switch are detected; that is, the suspected packet came through the interface with the NI-ID and through the layer 2 switch's port with the Port-ID. If any one of the bits is 0, the suspected packet was not stored with the pair of identifiers. Then, xDGA generates hash values by another pair of identifiers and checks the L2 digest table again. If all combinations of identifiers have been tested, but the true pair could not be detected in any trial, the suspected packet was not stored in the L2 digest table. Then, if another L2 digest table is suspected to record the packet signature in question, xDGA tests it.

To reduce the storage requirements, xDGA stores two identifiers in the L2 Bloom filter, though the running time of our searching algorithm is $O(n^2)$.

```
[ extracting identifiers in L2DT procedure ]


receive a suspected packet transferred from Traceback Manager

extract the packet signature PS from the suspected packet
check the time span through the suspected packet
  on the leaf router
search appropriate time L2DTes from ring-buffer for L2DT
for (number of L2DT to test){
  for (number of MAX_NIID_NUMBER){
    for (number of MAX_PORTID_NUMBER){
      make a hash input I by concatenating PS,
        an NI-ID N_i, and a Port-ID P_i
      for (each hash function (1 ≤ i ≤ k)){
        compute hash values H(I)_i
      }
      for (each hash value H(I)_i){
        search the bit indicated by H(I)_i on L2DT
        if (the bit ≡ 0){
          break the loop
        } else if (all bits indicated by hash values are 1){
          N_find = N_i, P_find = P_i
          write down N_find and P_find in syslog
          warn as finding the route of the suspected packet
          show the incoming interface N_find
            and the port P_find
          exit the procedure
        }
      }
      if (all Port-ID has been tested){
        break the loop
      }
    }
    if (all NI-ID has been tested){
      break the loop
    }
  }
  if (all suspected L2DT has been checked ){
    the suspected packet was not stored into any L2DT
    log the warning message about failure of L2 traceback
    exit the procedure
  }
}
```

Figure 4.6: Procedure to check identifiers on L2DT

4.6 Implementation of Layer 2 Extension of SPIE

We have implemented a PC-based prototype of L2-SPIE, based on an open source code of hash-based IP traceback, spie-1.0.90 [15], on FreeBSD-4.3.

We have added two components in SPIE and extended DGA to deal with L2 digest tables (Fig. 4.7). Fdbgetter is an agent which fetches FDB entries from layer 2 switches directly connected to the leaf router via SNMP and makes conversion tables. x_spiemem, a loadable kernel module, converts MAC addresses of an incoming packet to two identifiers and stores these identifiers in an L2 digest table. The packet storing processes by spiemem, which is a loadable kernel module to record packet signatures for the purpose of IP traceback, and the processes of recording identifiers by x_spiemem are independent of each other. Therefore, the Bloom filter's parameters for L2 digest tables and L3 digest tables can be set independently to achieve a different size or false positive rate.

For the hash functions of L2 digest tables, we reuse MD5, which SPIE computes for L3 digest tables. Therefore, different hash functions are simulated by a salt value seeded on the leaf router. The salt values for L2 digest tables are generated independently from the salt value for L3 digest tables.

xDGA on the leaf router checks L2 digest tables while testing L3 digest tables for IP traceback in parallel. The processes of examining L2 digest tables do not affect processes of SPIE's IP traceback because these processes are independent of each other.

The information from L2 digest tables is significant only for the administrator of the leaf router and should not be available to anyone else. Therefore, xDGA records the result of checking the L2 digest table in a log file and, shows a warning message on the standard output of the leaf router. xDGA does not send the information of identifiers to SCAR and/or STM.

4.7 Preliminary Evaluation

4.7.1 Memory Requirements

Our extension needs memory spaces for L2/ L3 digest tables and for conversion tables. The required memory space for L2 digest tables can be characterized by the duration of record storage and the accuracy of L2 digest tables measured by the number of

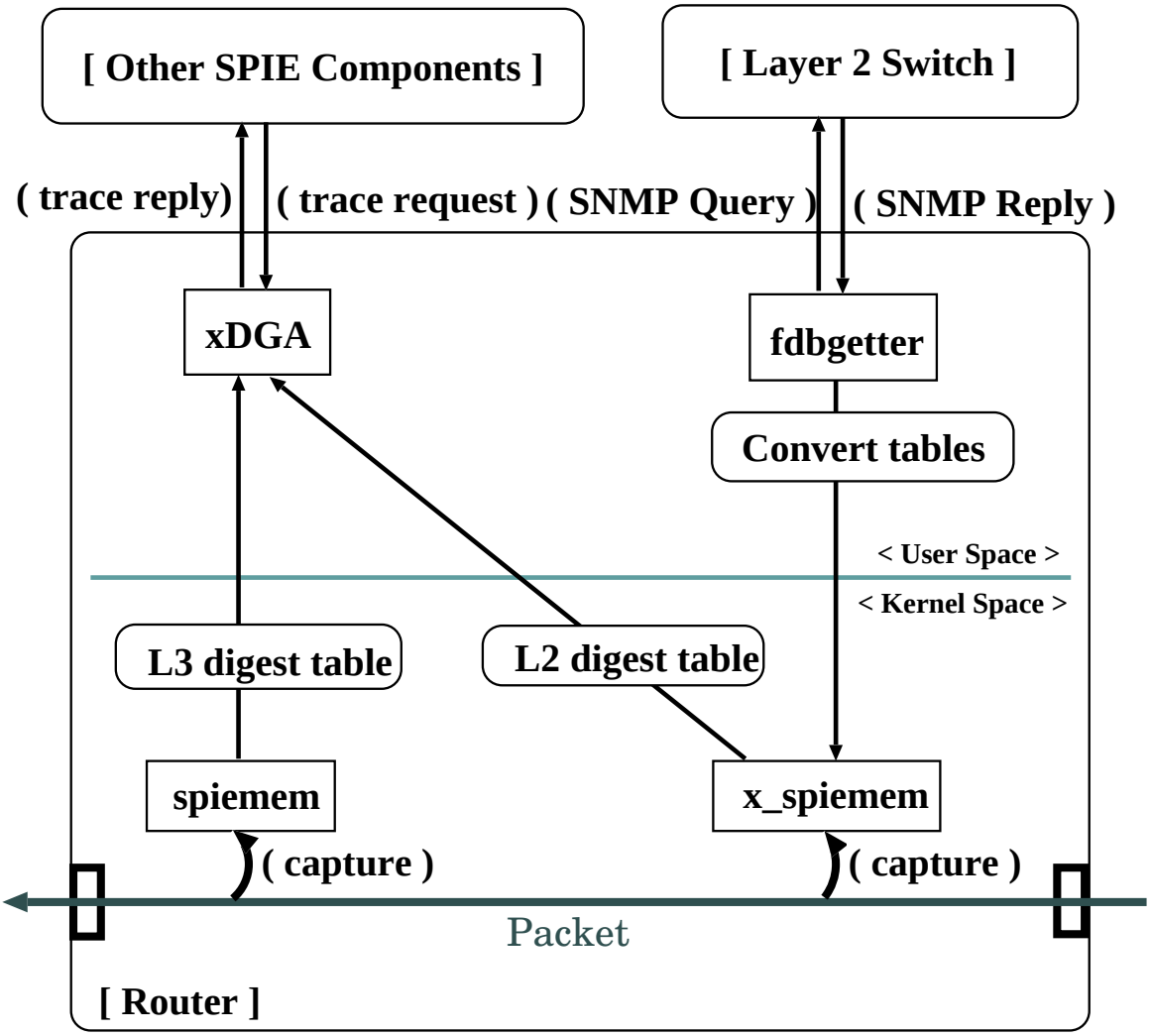


Figure 4.7: Components of the Layer-2 extension of SPIE

false positives. Meanwhile, the required memory space for L3 digest tables can be determined by these parameters for L3 digest tables. On the other hand, the required memory spaces for conversion tables are dominated by the number of network interfaces of a leaf router and affected by the sum of the number of FDB entries fetched from layer 2 switches connected to the leaf router.

Memory Requirements of the Conversion Table

In the implementation of L2-SPIE, an entry in the NI-ID table needs 16 bytes and an entry in a PortMAC table requires 12 bytes. The total memory size for conversion tables is dominated by the number of NI-ID table and the sum of FDB entries on each layer 2 switch. If a leaf router has three network interfaces, and each interface aggregates a class C (/24) subnet, the total memory size for conversion tables is a maximum of 9192 bytes.

Consider the case when a leaf router connects a BlackDiamond, which is today's High-end Layer-2 switch developed by Extreme Networks. A BlackDiamond can learn 256,000 entries on its FDB. Therefore, a PortMAC table for a BlackDiamond consumes a maximum of 3MB memory space. However, only rarely will the entry space of an FDB be full; the average size of the required memory for a PortMAC table will be much smaller.

Memory Requirements of the L2 Digest Table

The memory requirements for an L2 digest table are the same as those for an L3 digest table. The available memory size m with a capacity of storing n packets and a false positive rate P can be determined as

$$m = -n \cdot log(1/P)/ln(1/2) \approx 1.44n \cdot log(1/P).$$

A 2^{24} bits (2MB)-sized L2 digest table with a false positive rate of 0.092 can store 3,355,443 pairs of identifiers. Table 4.3 shows typical memory sizes and their capacity to store entries, when the number of hash functions is 3 and the false positive rate is 0.092. Table 4.3 shows the average bit rate stored in a Bloom filter when the average packet size is 1000 bits and the length of time required to use a Bloom filter is 10 seconds.

Table 4.3: Capacity of a Bloom filter

memory size	max num of entries	average bit rate
8 kB (2^{16} bit)	13,107	1 Mbps
128 kB (2^{20} bit)	209,715	20 Mbps
2MB (2^{24} bit)	3,355,443	335 Mbps
32MB (2^{28} bit)	53,687,091	5.3 Gbps
512MB (2^{32} bit)	858,993,459	85.8 Gbps

4.7.2 Preliminary Experiment

Experiment Environment

We have evaluated the prototype implementation of L2-SPIE on a testbed network to measure the decoding overhead on L2-SPIE which takes $O(n)$. Figure 4.8 shows the L3/L2 network topology of the testbed.

The experiment was carried out in the following situation: with only one xDGA leaf router; and a traceback manager STM.

On the Victim Net, there were two victim nodes (monitor agents): one received attack packets from e4 (connected to port 6 on FastIron) and e5 (connected to port 7) on the Attacker Net, and the other received attack packets from e6 (connected to port 8) and e8 (connected port 10) on the Attacker Net. Attack packets were TCP SYN packets transferred to the victim's TCP port 7777, generated by a customized the SYN flood program based on the libnet sample SYN flood program [94]. Each attack packet had a spoofed source IP address and a spoofed source TCP port. A sample monitor agent of spie-1.0.90 sent trace requests about all packets received by snort 1.9.0 beta4 [95]. In this experiment, the snort on each victim node was set as a host IDS to catch only attack packets on the victim node so that SPIE components traced only attack packets. Each attacker node sent attack packets at different time intervals (table 4.4).

In testbed topology, the network for exchanging traceback queries was separated from networks to exchange normal/attacking traffic; that is, Traceback Net transmitted only traceback messages (fig. 4.8).

The specification of the xDGA router is as follows: Pentium III 800MHz processor,

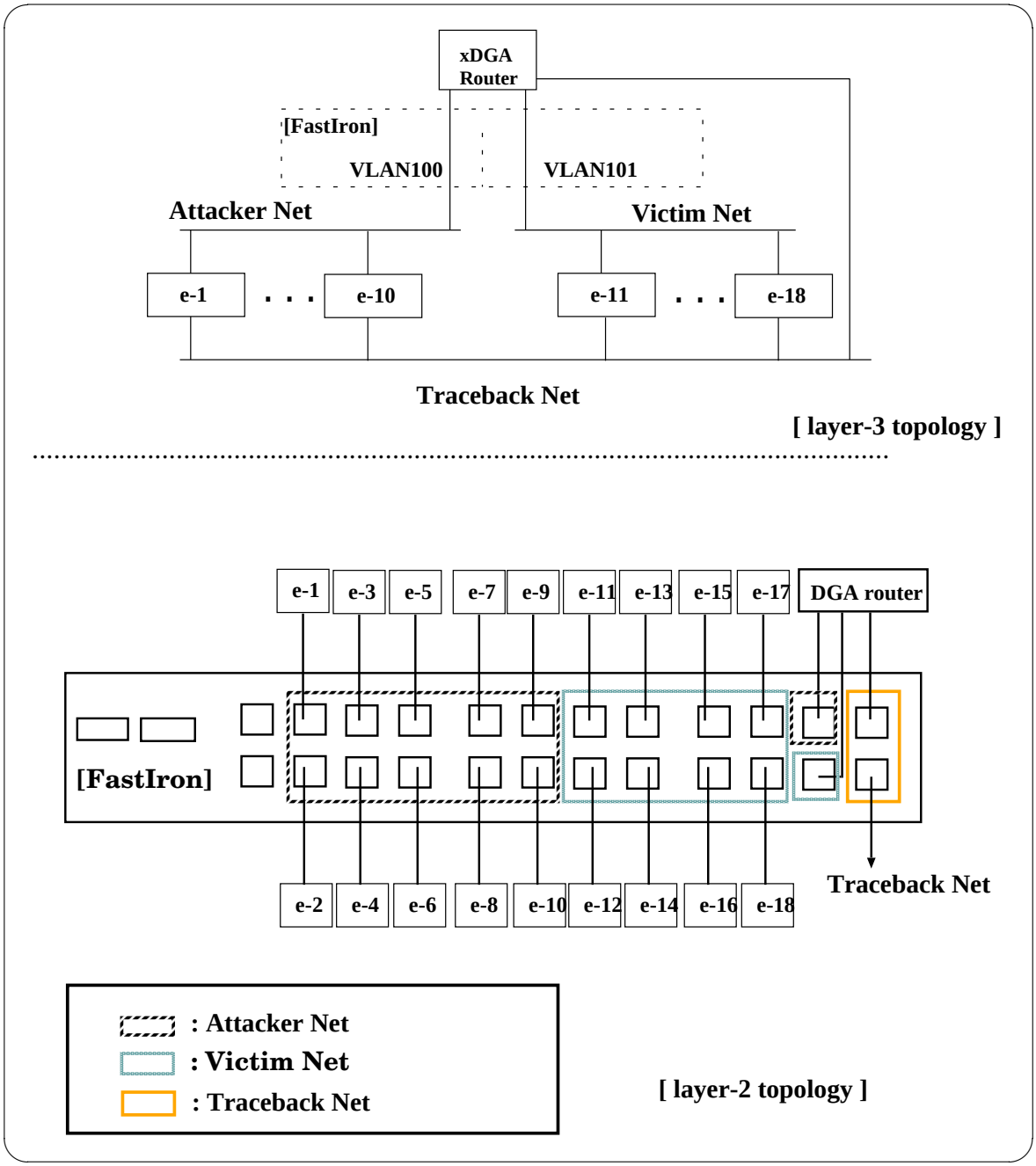


Figure 4.8: Testbed topology

Table 4.4: Time interval of each attacker node

name	interval
e4	3 (sec.)
e5	5 (sec.)
e8	7 (sec.)
e10	11 (sec.)

Table 4.5: Parameters of Bloom filters

size of hash values	24 bits
size of digest table	2MB (2^{24} bits)
length of time to refresh digest table	10 sec.
number of digest tables held on ring buffer	6
number of hash functions	3
capacity factor of Bloom filter	5

128MB sized memory, and four physical network interfaces. To build L2 networks, we used a layer 2 switch, Foundry's FastIron, firmware version 7.1.26T10. Each network exchanging normal/attacking traffic is connected to the xDGA router.

Parameters of the Bloom filter are described in Table 4.5. In this experiment, the values of parameters for L2 digest tables are set to the same set of parameters for L3 digest tables. (Parameters of Bloom filters for L2 digest tables can be set to different values from parameters for L3 digest tables.) Through calculation of parameters shown in Table 4.5, each L2/L3 digest table could store about 3.3 million packets at a 0.092 false positive rate, and could hold each digest table for 1 minute.

The time interval for fetching the FDB entries from the layer 2 switch was set to 60 seconds.

Table 4.6: Time spent to check a digest table

time	L3	L2
max (micro sec.)	70564	68474
min (micro sec.)	297	333
average (micro sec.)	381	1124
median (micro sec.)	333	1067

Experimental Result

In this experiment, each attacker node sent 2000 attack packets to the victim node, so STM and xDGA dealt with 8,000 packets altogether. As a result, all attack packets were found on the appropriate L3 and L2 digest tables.

Table 4.6 shows the time spent in finding the record of the suspected attack packet on the appropriate digest table. Each maximum time value is highly exceptional. In most cases, the time spent to find a packet record is within the range from 297 to 1704 microseconds on the L3 digest table, and from 333 to 2733 microseconds on the L2 digest table. Comparing the results for L3 and L2 digest tables, it was found that there is no critical performance overhead for searching for an appropriate ID pair on an L2 digest table in this experiment.

4.7.3 Summary

We have proposed IPBT method as a single packet traceback technique on a layer 2 network and a layer-2 extension of SPIE, which stores the layer 2 information of incoming packets on each leaf router. Using the layer 2 information, the port on a particular slot of the layer 2 switch that learned the incoming attack packet can be detected, which leads to a narrowing down of the existence range of an attacker node on a layer 2 network. Preliminary evaluation of a prototype implementation shows the potential usability of a layer 2 traceback.

In this chapter, we have proposed an extension approach of SPIE for the first step of a layer 2 traceback technique, and we have developed an algorithm to reduce the memory requirements for storing identifiers based on Bloom filters. The preliminary evaluation showed that all traceback trials found appropriate ID pairs with no critical overhead on the system in a small topology.

Of course, in addition to the SPIE enhanced router, L2-SPIE has the same problems with access rates on memory in the high speed network mentioned in Section 4.2.1. In future work, a more space- and time-efficient algorithm for IPBT will be explored.

Chapter 5

Conclusion

This dissertation addressed the problems faced by traceback techniques. The issues on the inter-domain include how to overcome the barriers on the network boundaries, how to reduce the overhead of the cooperation among ASes, how to provide the independence from a specific technique, and how to automate the inter-domain traceback operation. Unfortunately, existing proposals of inter-domain traceback architectures and techniques cannot overcome these issues. On the other hand, the issues of the intra-domain traceback are how to detect the location of an attacker node on a layer 2 network by the signature of a single packet and how to link the tracking on layer 3 networks and the tracking on layer 2 networks. Although there are several layer 2 traceback techniques, each has limitations which are caused by the aging of FDB or by the lack of binding technique between the layer 2 traceback and the layer 3 traceback.

InterTrack has been proposed in this dissertation as an interconnection architecture of traceback systems for the inter-domain traceback trials and L2-SPIE as an interconnection technique between the layer 2 traceback and the layer 3 traceback. These proposals were evaluated with prototype implementations.

These proposals can achieve an inter-domain single packet traceback system which can detect the location of attacker nodes on the layer 2 network without the violation of the operational policies of each AS.

5.1 Contributions

In an attempt to design the traceback architecture to overcome issues faced by existing proposals of traceback techniques, this dissertation makes the following specific contributions:

- The loose cooperation among ASes on the deployment traceback systems and on traceback operation that is based on the federated traceback architecture of InterTrack and on the independence of InterTrack from a specific traceback technique.

  The InterTrack architecture are designed for loosening the cooperation among ASes on the deployment of traceback systems. Dividing the operational boundaries of a traceback trial along with the network boundaries and by the phased-tracking approach allow each AS to operate a traceback trial according to AS's operational policy. Through the module components of InterTrack, ASes can choose traceback systems for internal use in accordance with the pros and cons of each system or the investment cost, regardless of the traceback systems on other ASes.

  And InterTrack messages are designed to conceal the sensitive information of ASes. The ITM trace reply message contains only the reverse AS path information, and it does not contain the detailed topologies of ASes; therefore, the leakage of sensitive information of each AS will not occured by replying a traceback trial.

- The same manner of traceback operation as the manner of other network operations which is brought by the phased-tracking approach of InterTrack.

  The phased-tracking approach was designed along with the network operation boundaries on the routing operation. By phasing an inter-domain traceback into four stages, ASes can authorize the users at each stage. ASes can also delegate the internal traceback operation to the organizations on the IGP sub-domains.

- The model of the border tracking technique for identifying the upstream neighbor ASes.

  Based on the assumption about the BGP-peering among ASes, we analyzed the variations of the AS status against a DDoS attack, and we modeled the border

tracking technique to aggregate the reverse path of an attack from the router-hop level to the AS-hop level. This model of the border tracking technique is independent from a specific traceback techniques. As a proof-of-concept program, we developed a sample BTM for a hash-digest-logging-based traceback product according to this model.

- The automated procedures on the inter-domain traceback through InterTrack. The procedures of a traceback trial on InterTrack was designed to automate the manual traceback over the network operation boundaries.

  The preliminary experiments of InterTrack showed that the average time of a trial on InterTrack was estimated as 16 seconds with a hash-digest-logging-based border tracking system in 9 AS hop length. Comparison with other inter-domain traceback techniques indicated that InterTrack will automate the manual traceback and shorten the time spent for a traceback trial as well as other traceback techniques do.

- The detailed internal inspection about the source of an attack packet that is produced by the characteristics of the IPBT method.

  The IPBT method in this dissertation can track back a single packet on a layer 2 network to the port to which an attacker node connects despite the source-spoofed characteristic of the IP address and the MAC address.

- An interconnection between a hash-digest-logging method on the layer 3 traceback and IPBT method on the layer 2 traceback provided by L2-SPIE. In this dissertation, a prototype implementation of L2-SPIE was developed.

  The evaluation result of the prototype implementation shows the average time of a traceback trial on L2-SPIE was about 1,500 micro-seconds.

- The foundation of the self-defending network architecture that would be provided by the cooperation among traceback systems, detection systems, and protection systems through InterTrack and L2-SPIE.

  InterTrack and L2-SPIE provide a multi-layers' traceback to detect the attacker nodes from the layer 3 traceback to the layer 2 traceback. InterTrack was also designed to cooperate with other detection systems and protection systems for the multi-layers' traceback and the attack mitigations.

InterTrack will realize the inter-domain traceback in the practical network operations and automate the communication among any countermeasures of DDoS attacks over the network operation boundaries in a practical way. InterTrack and L2-SPIE will also achieve the foundation of the traceability on the Internet.

5.2 Open Issues

Despite our best intentions, this dissertation raises many more issues than it addresses. We discuss several of the most interesting ones below, as well as possibilities for extensions to the InterTrack or the L2-SPIE architecture.

5.2.1 Policy Interface

InterTrack currently lacks an expressive policy interface. In particular, there is no way to describe a filter rule to refuse an ITM trace request message through some untrusted ASes. Therefore, refactoring the prototype implementation of InterTrack for the open sourced release is taking place at the time of this dissertation. In this refactor, AS path filter function is developed like the AS path filter on the BGP [96]. An ITM trace request message has an ITM path list which shows the ASes that forward the ITM trace request message. Hence, developing the AS path filter function is not difficult.

5.2.2 Binding Techniques for a Multi-Layers Traceback

Mentioned in Section 3.6.1, InterTrack aims to track the attack in multi-layers by the cooperation of detections systems and continuous trials of traceback in order to detect commander nodes or the PC of an attacker. For a multi-layer traceback, some binding techniques of traceback techniques on different layers are required. Our L2-SPIE is one of the binding methods to interconnect SPIE as a layer 3 traceback and IPBT as a layer 2 traceback. In order to interconnect layer 3 traceback techniques and FDB-checking methods, another binding technique, which records the pair of a source IP address and a source MAC address, is required.

Today, layer 2 networks has become more complex or hierarchical network with encapsulation techniques such as PPPoE [57], MPLS [97], IEEE 802.1Q-in-Q VLAN Tag Termination [98], or IEEE 802.1ah Mac in Mac encapsulation [99]. Binding techniques

between layer 3 traceback and layer 2 traceback on such hierarchical layer 2 network or layer 2 traceback techniques across the complex layer 2 networks are topics on this research area.

Studying auditing techniques, which can categorize different traffic or packets in the same attack sequence, is one of issues in order to track back the origin of a worm, step stones or bot nets.

5.2.3 Privacy Issues

In this dissertation, the privacy of the payload of an attack packet was ignored through the discussion in section 3.6.2. However, the privacy of the packet payload or the secret of the data for a query must be considered for a multi-layer traceback.

There is a trade-off between the secret of the data for a query and the choices of traceback systems on each domain. InterTrack conveys the data for a query, an issued packet data to be traced in other words, by an ITM Trace Request message to other domains. The data conveyed by InterTrack is used as an input data for several traceback systems. If the data for a query is encoded by some one-way hash function for the secret, traceback systems on each tracking stage will have to employ the decoding method; hence, achieving the secret of the data may limit the choices of traceback techniques or methods. Exploring a practical approach to solve the secret of the issued packet is the future work of this dissertation.

5.2.4 Traceback Operation

InterTrack just automates the procedures of the request on a traceback trial according to the assumptions and the model about ASes mentioned in Captor 3. We have to verify the appropriateness of the model through the actual operation or the feedback from operators.

To succeed in a traceback trial with InterTrack or to confirm the result of a traceback trial, the cooperation of ISP operators is required. Some specialized instant messaging tool or bulletin board tool based on the certificate of InterTrack will encourage communication among ISP operators. Such supportive tools will also ensure the operation of InterTrack or the attack protection through InterTrack.

5.2.5 Non-spoofable Network Architecture

If every layer 2 switch or edge router does not forward any source-spoofed Ethernet frames or packets, then, the operation on the attack mitigation would become easy because the source IP address of a packet always indicates the true sender node. In addition to ingress filtering techniques, some authentication mechanism or auditing mechanism is required in order to discard unauthorized / spoofed ethernet frames on leaf subnets. 802.1x [100] or S-ARP [101] are authentication mechanisms using the Public Key Infrastructure (PKI) techniques to identify and authorize connected end nodes. Although these authentication mechanisms are useful for grasping the connected nodes or eliminating unauthorized nodes on leaf subnets, detecting source-spoofed packets or Ethernet frames from authorized nodes is difficult. Furthermore, using these PKI authentication mechanisms on an open network such as the wireless network access points on stations, or on coffee shops is also difficult.

In addition to these authentication mechanisms, it is necessary to develop a lightweight auditing mechanism which runs on a layer 2 switch and checks the inconsistency among the source IP / MAC address of an Ethernet frame and the IP / MAC address of the sender node. Mentioned in section 4.3, a layer 2 switch always learns a new MAC address and its incoming port on the FDB regardless of spoofed characteristics of ethernet frames or of IP packets. If a layer 2 switch could check the true sender node when the layer 2 switch receives or learns a new set of the source MAC address, the source IP address and the incoming port, the layer 2 switch could record the location of a suspected end node or might filter spoofed packets before the layer 2 switch forward these packets to other networks. The challenge on this checking mechanism is how to achieve the checking algorithm with keeping the performance of a layer 2 switch such as throughput or forwarding speed. L2-SPIE or other FDB-checking methods feed practical tips to researchers on developing such auditing mechanism.

5.2.6 Self-defending Network Architecture

Self-defending or DoS limiting network architecture is a current topic on network security [72, 102]. This dissertation contributed InterTrack as a media to interconnect traceback systems, detection systems, and/or protection systems for the self-defending. Although InterTrack would provide the foundation of the cooperation among any kinds of DoS/DDoS countermeasures, there are issues to be overcome for achieving an auto-

mated self-defending network architecture. The most interesting issues to construct a self-defending Internet architecture as follows:

- How to describe a security policy which can be translated as a configuration of each DoS/DDoS countermeasure.
- How to apply a security policy into each DoS/DDoS countermeasure.
- What manner or sequence DoS/DDoS countermeasures cooperate each other in accordance with a security policy.

A well-considered method on making and applying a security policy is required to automate the process of defending against DoS/DDoS attacks. eXtensible Access Control Markup Language (XACML) is an XML schema for an extensible access-control policy, and it might achieve rules of describing a security policy and a well-considered manner of automating the defending process. Operating InterTrack with such policy description and well considered procedures would expedite the cooperation among ASes on defending network attacks. Constructing a self-defending network architecture is an important part of this dissertation's future work. At the initial stages of constructing a self-defending network architecture, the detailed procedures of a multi-layer traceback must be studied.

References

[1] J. Mirkovic and P. Reiher, "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms," *ACM Computer Communications Review*, vol. 34, no. 2, pp. 39–54, Apr. 2004.

[2] D. McPherson and C. Labovitz, "WorldWide ISP Security Report," Arbor Networks, Sep. 2005. [Online]. Available: http://www.arbor.net/downloads/Arbor_Worldwide_ISP_Security_Report.pdf

[3] P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing," RFC 2827 (Best Current Practice), May 2000, updated by RFC 3704. [Online]. Available: http://www.ietf.org/rfc/rfc2827.txt

[4] F. Baker and P. Savola, "Ingress Filtering for Multihomed Networks," RFC 3704 (Best Current Practice), Mar. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3704.txt

[5] Cisco Systems, Inc., "Unicast Reverse Path Forwarding Enhancements." [Online]. Available: http://www.cisco.com/en/US/products/sw/iosswrel/ps1834/products_feature_guide09186a008007fffd.html

[6] R. Beverly and S. Bauer, "Spoofer project." [Online]. Available: http://spoofer.csail.mit.edu/

[7] ——, "The spoofer project: Inferring the extent of source address filtering on the internet," in *Proceedings of USENIX Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI)*, July 2005, pp. 53–59. [Online]. Available: http://www.usenix.org/events/sruti05/tech/beverly.html

[8] A. Belenky and N. Ansari, "On IP Traceback," *IEEE Communications Magazine*, vol. 41, no. 7, pp. 142–153, July 2003.

[9] T. Battles, D. McPherson, and C. Morrow, "Customer-triggered real-time blackholes," NANOG, Tech. Rep., Feb 2004.

[10] B. R. Greene and D. McPherson, "Sink holes: A swiss army knife isp security toolversion 1.8," NANOG 28, Tech. Rep., June 2003. [Online]. Available: http://ipmon.sprint.com/pubs_trs/trs/RR04-ATL-013177.pdf

[11] Cisco Systems Inc., "Remotely triggered black hole filtering - destination based and source based," Cisco Systems Inc., Tech. Rep., Feb 2005. [Online]. Available: www.cisco.com/warp/public/732/Tech/security/docs/blackhole.pdf

[12] B. Claise, "Cisco Systems NetFlow Services Export Version 9," RFC 3954 (Informational), Oct. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3954.txt

[13] P. Phaal, S. Panchen, and N. McKee, "InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks," RFC 3176 (Informational), Sept. 2001. [Online]. Available: http://www.ietf.org/rfc/rfc3176.txt

[14] Arbor Networks, "Peakflow SP." [Online]. Available: http://www.arbornetworks.com/products_sp.php

[15] BBN Technologies, "Source Path Isolation Engine (SPIE)," http://www.ir.bbn.com/projects/SPIE/index.html.

[16] Institute of Applied Internet Technology Inc., "PAFFI: PAcket Footmark FInder." [Online]. Available: http://www.netstar.co.jp/products/PAFFI/index.html

[17] Cyber Solutions Inc., "Packetchaser (in japanese)." [Online]. Available: http://netskate.cysol.co.jp/products/packetchaser/index.html

[18] Cisco Systems, Inc., "Catalyst 6500 Series Command Reference, 8.2 - fotmat to ping - l2trace." [Online]. Avail-

able: http://www.cisco.com/en/US/products/hw/switches/ps708/products_command_reference_chapter09186a00801dd5dd.html#wp1030529

[19] ——, "Cisco Works Small Network Management Solution Version 1.5." [Online]. Available: http://www.cisco.com/en/US/products/sw/cscowork/ps2408/prod_brochure09186a00801c0a43.html

[20] Foundry Networks, Inc, "IronView Network Manager Features." [Online]. Available: http://www.foundrynet.com/products/networkman/ironview/features.html

[21] Extreme Networks, "EPICenter Asset Discovery Tool." [Online]. Available: http://www.extremenetworks.com/libraries/prodpdfs/products/epicenter_ADT.asp

[22] Aruba Networks, "Aruba Tech Briefs." [Online]. Available: http://www.arubanetworks.com/technology/techbriefs.php

[23] M. Oe, H. Hazeyama, S. Yamamoto, and S. Shirahata, "An implementation and verification of ieee 802.11 wireless network management system," *Electronics and Communications in Japan (Part I: Communications)*, vol. 88, no. 12, pp. 20–28, June 2005.

[24] G. Sager, "Security fun with OCxmon and cflowd," http://www.caida.org/projects/ngi/content/security /1198/, Internet 2 working Group Meeting, Nov. 1998.

[25] R. Stone, "CenterTrack: an IP overlay network for tracking DoS floods," in *Proceedings of 9th USENIX Security Symposium '00, Denver, USA*, Aug. 2000.

[26] H. Burch and B. Cheswick, "Tracing anonymous packets to their approximate source," in *Proceedings of 14th USENIX Systems Administration Conference '00, New Orleans, Louisiana, USA*, Dec. 2000.

[27] S. Bellovin, M. Leech, and T. Taylor, "ICMP traceback message," Feb. 2003, IETF, Internet Draft, draft-ietf-itrace-04.txt.

[28] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical network support for IP traceback," in *Proceedings of ACM SIGCOMM'00*, Aug. 2000, pp. 295–306.

[29] A. C. Snoren, C. Partridge, L. A. Sanches, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Stayer, "Single-packet IP traceback," *ACM Transactions on Networking*, vol. 10, no. 6, pp. 119–137, Dec. 2002.

[30] J. Ioannidis and S. M. Bellovin, "Implementing Pushback: Router-based defense against DDoS attacks," in *Proceedings of Network and Distributed System Security Symposium, Catamaran Resort Hotel San Diego, California 6-8 February 2002*. 1775 Wiehle Ave., Suite 102, Reston, VA 20190: The Internet Society, Feb. 2002.

[31] T. Baba and S. Matsuda, "Tracing network attacks to their sources," *IEEE Internet Computing*, vol. 13, no. 7, pp. 422–426, Mar 2002.

[32] R. Ramanujan, M. Kaddoura, J.Wu, K. Millikin, D. Harper, and D. Baca, "Organic techniques for protecting virtual private network (vpn) services from access link flooding attacks"," in *Proceedings of IEEE/IEE Conference on Networking (Networks 2002)*,, 2002.

[33] M. Oe, Y. Kadobayashi, and S. Yamaguchi, "An implementation of a hierarchical IP traceback architecture," in *Proceedings of IPv6 Workshop, SAINT 2003, Orland, USA*, Jan. 2003.

[34] V. Paruchuri, A. Durresi, L. Barolli, R. Kannan, and S. Ivengar, "Efficient and secure autonomous system based traceback," *Journal of Interconnection Networks (JOIN)*, vol. 5, no. 2, pp. 151–164, June 2004.

[35] J. Li, M. Sung, J. Xu, and L. Li, "Large-scale ip traceback in high-speed internet: Practical techniques and theoretical foundation." in *Proceedings of IEEE Symposium on Security and Privacy*, 2004, pp. 115–129.

[36] C. Jin, H. Wang, and K. G. Shin, "Hop-count filtering: an effective defense against spoofed ddos traffic." in *Proceedings of ACM Conference on Computer and Communications Security*, 2003, pp. 30–41.

[37] Hewlett-Packard Development Company, L.P., "Network Node Manager advanced edition." [Online]. Available: http://www.managementsoftware.hp.com/products/nnm/index.html

[38] Fluke Networks, "OptiView Integrated Network Analyzer." [Online]. Available: http://www.flukenetworks.com/us/LAN/Handheld+Testers/Optiview.htm

[39] K. McCloghrie and M. Rose, "Management Information Base for Network Management of TCP/IP-based internets:MIB-II," RFC 1213 (Standard), Mar. 1991, updated by RFCs 2011, 2012, 2013. [Online]. Available: http://www.ietf.org/rfc/rfc1213.txt

[40] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, July 1970.

[41] L. Wen, J. Wu, and K. Xu, "Overlay logging: An ip traceback scheme in mpls network." in *Proceedings of ICN*, 2005, pp. 75–82.

[42] N. Fischbach, "MPLS-based traffic shunt," RIPE46, Tech. Rep., Sep. 2003. [Online]. Available: www.securite.org/presentations/ripe46/COLT-RIPE46-NF-MPLS-TrafficShunt-v1.pdf

[43] T. Hamano, R. Suzuki, T. Ikegawa, and H. Ichikawa, "Redirection based defense mechanism against flood-type attacks for a large-scale isp network," in *Proceedings of 10-th Asia-Paciffic Conference on Communications APCC'04*, Aug. 2004.

[44] C. T. Sharad Agarwal, Travis Dawson, "Ddos mitigation via regional cleaning centers," SPRINT ATL RESEARCH REPORT RR04-ATL-013177, Tech. Rep., January 2004. [Online]. Available: http://ipmon.sprint.com/pubs_trs/trs/RR04-ATL-013177.pdf

[45] F. Kastenholz, "The Definitions of Managed Objects for the Bridge Network Control Protocol of the Point-to-Point Protocol," RFC 1474 (Proposed Standard), June 1993. [Online]. Available: http://www.ietf.org/rfc/rfc1474.txt

[46] TCPDUMP ORG, "The Libpcap library." [Online]. Available: http://www.tcpdump.org/

[47] S. F. Wu, W. Huang, D. Massey, A. Mankin, C. L. Wu, X. L. Zhao, and L. Zhang, "Intention-driven ICMP trace-back," Nov. 2001, IETF, Internet Draft, draft-ietf-itrace-intention-00.txt.

[48] T. Yamada, "Active traceback protocol," Oct. 2002, IETF, Internet Draft, draft-yamada-active-trace-00.txt.

[49] Y. Sawai, M. Oe, K. Iida, and Y. Kadobayashi, "Performance evaluation of inter-domain IP traceback," in *Proceesings of ICT'03, Tahiti*, Feb. 2003.

[50] D. X. Song and A. Perrig, "Advanced and authenticated marking schemes for IP traceback," in *Proceedings of IEEE Infocomm 2001*, 2001.

[51] D. Dean, M. Franklin, and A. Stubblefield, "An algebraic approach to IP traceback," *ACM Transactions on Information and System Security*, vol. 5, no. 2, pp. 119–137, 2002.

[52] M. Waldvogel, "GOSSIB vs. IP traceback rumors," in *Proceedings of 18th Annual Computer Security Applications Conference (ACSAC 2002)*, Dec. 2002.

[53] C. Shannon, D. Moore, and k claffy, "Characteristics of fragmented ip traffic on internet links," in *Proceedings of RIPE NCC a workshop on Passive and Active Measurements (PAM2001)*, Apr. 2001.

[54] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina, "Generic Routing Encapsulation (GRE)," RFC 2784 (Proposed Standard), Mar. 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2784.txt

[55] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, and B. Palter, "Layer Two Tunneling Protocol "L2TP"," RFC 2661 (Proposed Standard), Aug. 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2661.txt

[56] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol," RFC 2401 (Proposed Standard), Nov. 1998, updated by RFC 3168. [Online]. Available: http://www.ietf.org/rfc/rfc2401.txt

[57] L. Mamakos, K. Lidl, J. Evarts, D. Carrel, D. Simone, and R. Wheeler, "A Method for Transmitting PPP Over Ethernet (PPPoE)," RFC 2516 (Informational), Feb. 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2516.txt

[58] H. C. J. Lee, M. Ma, V. L. L. Thing, and Y. Xu, "On the issues of ip traceback for ipv6 and mobile ipv6," in *Proceedings of IEEE Symposium on Computers and Communications - ISCC'2003*, June 2003.

[59] A. Durresi, V. Paruchuri, L. Barolli, R. Kannan, and S. S. Iyengar, "Efficient and secure autonomous system based traceback." *Journal of Interconnection Networks*, vol. 5, no. 2, pp. 151–164, 2004.

[60] S. Floyd, S. Bellovin, J. Ioannidis, K. Kompella, R. Mahajan, and V. Paxson, "Pushback messages for controlling aggregates in the network," July 2001, iETF, Internet Draft, draft-floyd-pushback-messages-00.txt.

[61] K. Shanmugasundaram, H. Brönnimann, and N. D. Memon, "Payload attribution via hierarchical bloom filters." in *ACM Conference on Computer and Communications Security*, 2004, pp. 31–41.

[62] H. Song, S. Dharmapurikar, J. Turner, and J. Lockwood, "Fast hash table lookup using extended bloom filter: an aid to network processing," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 181–192, 2005.

[63] J. Li, M. Sung, J. Xu, and L. Li, "Large-scale ip traceback in high-speed internet: Practical techniques and theoretical foundation." in *Proceedings of IEEE Symposium on Security and Privacy*, 2004, pp. 115–129.

[64] C. Gong, T. Le, T. Korkmaz, and K. Sarac, "Single packet ip traceback in as-level partial deployment scenario," in *Proceedings of IEEE GLOBECOM 2005*, Nov. 2005.

[65] C. Partridge, C. Jones, D. Waitzman, and A. Snoeren, "New protocols to support internet traceback," Nov. 2001, IETF, Internet Draft, draft-partridge-ippt-discuss-00.txt.

[66] G. M. Keeni and Y. Kuwata, "An architecture for IP packet tracing," Oct. 2004, IETF, Internet Draft, draft-glenn-ippt-arch-01.txt.

[67] K. M. Moriarty, "Incident Handling: Real-time Inter-network Defense," Nov. 2005, IETF, Internet Draft, draft-ietf-inch-rid-05.txt.

[68] M. Oe, "A hierarchical architecture for IP traceback," Jul. 2002, IETF, a presentation in IPPT BoF at 54th IETF meeting. [Online]. Available: http://iplab.naist.jp/research/traceback/ippt-naist-ietf54.pdf

[69] R. Danyliw, J. Meijer, and Y. Demchenko, "The incident object description exchange format data model and xml implementation," Nov. 2005, IETF, draft-ietf-inch-iodef-05.txt.

[70] T. Kai, A. Nagashima, H. Nakatani, N. Fukuda, S. Hiroshi, A. Hashiguchi, and KatsujiTsukamoto, "Three classes based model of traceback system between ass," Feb. 2004, presentation on INCH Working Group session in 59th IETF meeting. [Online]. Available: http://www3.ietf.org/proceedings/04mar/slides/inch-2.pdf

[71] T. Kai, A. Nagashima, H. Nakatani, NaohiroFukuda, S. Hiroshi, A. Hashiguchi, T. Suzuki, and KatsujiTsukamoto, "Rid implementation report," Nov. 2004, presentation on INCH Working Group session in 61th IETF meeting. [Online]. Available: http://www1.ietf.org/proceedings_new/04nov/slides/inch-4.pdf

[72] Cisco Sytems Inc., "Cisco self-defending networks integrated protection against top security challenges." [Online]. Available: http://www.cisco.com/application/pdf/en/us/guest/netsol/ns413/c643/cdccont_0900aecd8024d539.pdf

[73] nCircle Network Security, Inc., "ntellect." [Online]. Available: http://www.ncircle.com/index.php?s=products_ntellect

[74] A. Conta and S. Deering, "Generic Packet Tunneling in IPv6 Specification," RFC 2473 (Proposed Standard), Dec. 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2473.txt

[75] B. Carpenter and C. Jung, "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels," RFC 2529 (Proposed Standard), Mar. 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2529.txt

[76] P. Srisuresh and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)," RFC 3022 (Informational), Jan. 2001. [Online]. Available: http://www.ietf.org/rfc/rfc3022.txt

[77] Microsoft Corporation, "Windows Server System - Internet Protocol Version 6." [Online]. Available: http://www.microsoft.com/windowsserver2003/technologies/ipv6/default.mspx

[78] Apple Computer, Inc., "MAC OS X Man Pages - ip6 – Enable or disable IPv6 on active interfaces." [Online]. Available: http://developer.apple.com/documentation/Darwin/Reference/ManPages/man8/ip6.8.html

[79] US-Cert, "Malware tunneling in ipv6," US-Cert, Tech. Rep., May 2005. [Online]. Available: http://www.us-cert.gov/reading_room/IPv6Malware-Tunneling.pdf

[80] S. Kent and R. Atkinson, "IP Encapsulating Security Payload (ESP)," RFC 2406 (Proposed Standard), Nov. 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2406.txt

[81] The Team Cymru, "The team cymru BGP data page." [Online]. Available: http://www.cymru.com/BGP/index.html

[82] G. Huston, "BGP Table Data." [Online]. Available: http://bgp.potaroo.net/index-bgp.html

[83] H. Hazeyama, M. Oe, and Y. Kadobayashi, "A layer-2 extension to hash-based IP traceback," *IEICE Transactions on Information and Systems*, vol. E86-D, no. 11, pp. 2325–2333, November 2003.

[84] Geoff Huston, "APNIC Trial of Certification of IP Addresses and ASes," Oct. 2005. [Online]. Available: http://www.ripe.net/ripe/meetings/ripe-51/presentations/pdf/ripe51-address-certificate.pdf

[85] C. Lynn, S. Kent, and K. Seo, "X.509 Extensions for IP Addresses and AS Identifiers," RFC 3779 (Proposed Standard), June 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3779.txt

[86] BBN Technologies, "Secure BGP Project (S-BGP)." [Online]. Available: http://www.net-tech.bbn.com/sbgp/sbgp-index.html

[87] R. White, "Securing BGP Through Secure Origin BGP." [Online]. Available: http://www.cisco.com/web/about/ac123/ac147/ac174/ac236/about_cisco_ipj_archive_article09186a00801c5a9b.html

[88] Daniel Veillard, "The XML C parser and toolkit of Gnome libxml." [Online]. Available: http://xmlsoft.org/

[89] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616 (Draft Standard), June 1999, updated by RFC 2817. [Online]. Available: http://www.ietf.org/rfc/rfc2616.txt

[90] VMware Inc., "Vmware workstation." [Online]. Available: http://www.vmware.com/products/ws/

[91] V. L. L. Thing, H. C. J. Lee, M. Sloman, and J. Zhou, "Enhanced icmp traceback with cumulative path," in *Proceedings of 61st IEEE Vehicular Technology Conference*, May 2005.

[92] L. F. Cao, J. Almeida, and A. Z. Broder, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol," http://www.cs.wisc.edu/~cao/papers/summary-cache/share.html.

[93] E. Decker, P. Langille, A. Rijsinghani, and K. McCloghrie, "Definitions of Managed Objects for Bridges," RFC 1286 (Proposed Standard), Dec. 1991, obsoleted by RFCs 1493, 1525. [Online]. Available: http://www.ietf.org/rfc/rfc1286.txt

[94] M. D. Schiffman, "The libnet library." [Online]. Available: http://libnet.sourceforge.net/

[95] B. Caswell and M. Roesch, "Snort." [Online]. Available: http://www.snort.org/

[96] Y. Rekhter and P. Gross, "Application of the Border Gateway Protocol in the Internet," RFC 1772 (Draft Standard), Mar. 1995. [Online]. Available: http://www.ietf.org/rfc/rfc1772.txt

[97] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," RFC 3031 (Proposed Standard), Jan. 2001. [Online]. Available: http://www.ietf.org/rfc/rfc3031.txt

[98] Cisco Systems, Inc., "IEEE 802.1Q-in-Q VLAN Tag Termination." [Online]. Available: http://www.cisco.com/en/US/products/sw/iosswrel/ps5207/products_feature_guide09186a00801f0f4a.html#wp1051003

[99] Institute of Electrical and Electronics Engineers, Inc., "802.1ah - Provider Backbone Bridges." [Online]. Available: http://www.ieee802.org/1/pages/802.1ah.html

[100] ——, "802.1X - Port Based Network Access Control." [Online]. Available: http://www.ieee802.org/1/pages/802.1x.html

[101] M. G. Gouda and C.-T. Huang, "A secure address resolution protocol." *Computer Networks*, vol. 41, no. 1, pp. 57–71, 2003.

[102] X. Yang, D. Wetherall, and T. Anderson, "A DoS-limiting Network Architecture," in *Proceedings of ACM SIGCOMM'05*, Aug. 2005.

Appendix A

List of Publications

A.1 Journal

1–1. Hiroaki Hazeyama, Masafumi Oe and Youki Kadobayashi, "A Layer-2 Extension to Hash-based IP Traceback", IEICE Transactions on Information and Systems, Vol. E86-D, No 11, pp. 2325–2333, November 2002.

1–2. 大江 将史, 櫨山 寛章, 山本 成一, 白畑 真: "IEEE802.11 ワイヤレスネットワーク管理システムの構築と検証", 電子情報通信学会論文誌, Vol. J87-B, No. 10, pp. 1607–1615, 2004 年 10 月.

1–3. Masafumi Oe, Hiroaki Hazeyama, Seiichi Yamamoto and Sin Shirahata, "An implementation and verification of IEEE 802.11 wireless network management system", Electronics and Communications in Japan, Vol.88-12, pp. 20-28, June 2005.

1–4. Takuji Iimura, Hiroaki Hazeyama and Youki Kadobayashi, "Distributed Scalable Multi-player Online Game Servers on Peer-to-Peer Networks", 情報処理学会論文誌 Vol.46, No. 2, pp.276–391,2005 年 2 月.

A.2 International Conference

2–1. Hiroaki Hazeyama, Masafumi Oe and Youki Kadobayashi, "A Layer-2 Extension to Hash-based IP Traceback", In *1st International Forum on Information and Computer Technology (IFICT)*, Hamamatsu, Shizuoka, Japan, January 2003.

2–2. Hiroaki Hazeyama, Youki Kadobayashi, Masafumi Oe and Ryo Kaizaki, "Inter-Track: A federation of IP traceback systems across borders of network operation domains", In *Proceedings of Annual Computer Security Applications Conference, Technology Blitz Session*, Tucson, Arizona, U.S.A., December 2005.

2–3. Hiroaki Hazeyama, Youki Kadobayashi, Miyamoto Daisuke and Masafumi Oe, "An Autonomous Architecture for Inter-Domain Attack Traceback", In *Proceedings of In 11th IEEE Symposium on Computers and Communications (ISCC '06)*, Pula-Cagliari, Sardinia, Italy, June 2006 (submitted).

2–4. Takuji Iimura, Hiroaki Hazeyama and Youki Kadobayashi, "Zoned Federation of Game Servers: A Peer-to-Peer Approach to Scalable Multi-Player Online Games", In *Proceedings of ACM SIGCOMM Workshop Network and System Support for Games (NetGames 2004)*, Portland, Oregon, U.S.A., August 2004.

2–5. Daisuke Miyamoto and Hiroaki Hazeyama and Youki Kadobayashi, "SPS: a simple filtering algorithm to thwart phishing attacks", In *ASIAN INTERNET ENGINEERING CONFERENCE(AINTEC) 2005*, Bangkok, Thailand, December 2005.

2–6. Mio Suzuki and Hiroaki Hazeyama and Youki Kadobayashi, "Expediting experiments across testbeds with AnyBed: a testbed-independent topology configuration tool", In *2nd International IEEE/Create-Net Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCOM '06)*, Barcelona, Spain, March 2006 (to be appeared).

A.3 Technical Report

3–1. 櫨山寛章, 大江将史, 門林雄基 "MAC トレースバック:Hash-Based IP トレースバック拡張方式の提案", 電子情報通信学会 インターネットアーキテクチャ研究会, 2002 年 7 月.

3–2. 大江 将史, 櫨山 寛章, 門林 雄基: "トレースバックシステムの相互接続アーキテクチャの提案", 2005 年暗号と情報セキュリティシンポジウム予稿集, Vol. III, pp. 1549–1554, 2005 年 1 月.

3–3. 岡田 行央, 飯村 卓司, 櫨山 寛章, 門林 雄基, 山口 英: “ソーシャルインターネットワークの提案”, 電子情報通信学会技術研究報告 (MoMuC2004-88), (IA2005-19(2005-01)), pp.1–6, 2004 年 6 月.

3–4. 鈴木 未央, 櫨山 寛章, 門林 雄基: “AnyBed: クラスタ環境に依存しない実験ネットワーク構築支援機構の設計と実装”, 情報処理学会研究報告, 2004-OS-96-(17), pp.121–128, 2005 年 1 月.

3–5. 大宅 裕史, 櫨山 寛章, 門林 雄基: “ワーム拡散の遅延を目的とした検知・遮断機構の提案”, 情報処理学会研究報告, 2005-CSEC-28-(5), 2005 年 3 月.

3–6. 宮本 大輔, 鈴木 未央, 櫨山 寛章, 門林 雄基: “フィッシング攻撃対策ツールの有効性評価のためのスキャナの提案”, 暗号と情報セキュリティシンポジウム（SCIS2006）, 2006 年 1 月.