2012 32nd International Conference on Distributed Computing Systems Workshops

A Flow-Based Traceback Scheme on an AS-Level Overlay Network

Hongcheng Tian, Jun Bi, and Peiyao Xiao
Network Research Center, Tsinghua University
Department of Computer Science, Tsinghua University
Tsinghua National Laboratory for Information Science and Technology (TNList)
Beijing, China
{tianhc, xiaopy}@netarchlab.tsinghua.edu.cn, junbi@tsinghua.edu.cn

***Abstract*—IP traceback can be used to find the origins and paths of attacking traffic. However, so far, no Internet-level IP traceback system has ever been deployed because of deployment difficulties. In this paper, we present a flow-based traceback scheme on an AS-level overlay network (EasyTrace). In EasyTrace, it is not necessary to deploy any dedicated traceback software and hardware at routers, and an AS-level overlay network is built for incremental deployment. We theoretically analyze the quantitative relation among the probability that a flow is successfully traced back various AS-level hop number, independently sampling probability, and the packet number that the attacking flow comprises.**

Keywords- IP traceback, sampling, overlay network

I. INTRODUCTION

Distributed Denial of Service (DDoS) attacks continue to pose major threats to the Internet. Attackers can launch attacking traffic from various locations in the Internet to exhaust the bandwidth or computing resources at the victim. Attackers often forge source addresses to escape detection, such as SYN flooding, DNS amplification, Smurf, etc.

The objective of IP traceback [1] is to find the origins and attacking paths of malicious traffic. IP traceback is executed with the assistance of a series of routers. In addition, IP traceback can also collect statistics for packets' forwarding path(s) in the Internet in order to optimize router configuration, which benefits the research in the area of traffic engineering.

Most IP traceback approaches are difficult to be deployed in the Internet, because dedicated software or hardware needs to be deployed at routers, or most methods are difficult to be incrementally deployed. As far as the authors know, there is no Internet-level IP traceback system that is currently deployed.

In this paper, we propose an incrementally deployable IP traceback system that is based on sampled flows (EasyTrace). EasyTrace uses existing xFlow (sFlow, NetFlow and IPFIX) function and BGP information to implement traceback, instead of deploying any traceback software or hardware at routers. EasyTrace builds an AS-level overlay network among EasyTrace-enabled ASes by the upstream logical neighbor discovering in order to support the incremental deployment. It should be emphasized that, EasyTrace can confirm the ingress interface(s) of the BGP router(s) through which some attacking flows enter the EasyTrace-enabled AS. Theoretical analyses show that the probability that a flow is successfully traced back various AS-level hop number quantitatively depends on two factors: independently sampling probability, and the number of packets that the flow comprises.

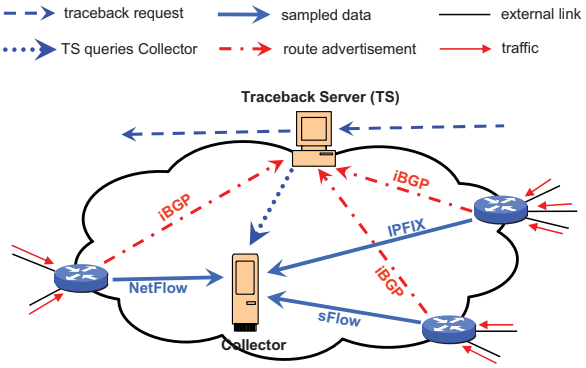

Figure 1. Intra-AS infrastructure in EasyTrace, consisting of a Traceback Server (TS), one or more Collectors.

The rest of the paper is organized as follows: In Section II, we describe the related work. Section III presents the EasyTrace system. Section IV evaluates its performance. And in Section V, the conclusions and our future work are given.

II. RELATED WORK

Researchers have proposed various approaches to trace attacking traffic back to source(s) and identify attacking path(s).

A. General Background

IP traceback schemes can be categorized as six types: Link testing (input debugging and controlled flooding), packet marking, ICMP traceback, logging-based schemes, hybrid IP traceback approaches, and overlay network for IP traceback.

Link testing includes input debugging and controlled flooding [2]. Input debugging takes advantage of a function of routers, which can identify the ingress link of attacking packets with a certain signature. The attack signatures are extracted by the victim from attacking packets and are sent

1545-0678/12 $26.00 © 2012 IEEE
DOI 10.1109/ICDCSW.2012.49

to the victim's upstream router, where the ingress port of attacking packets can be identified. This process is repeated recursively hop by hop. Controlled flooding is to flood links of a router with large bursts of traffic and observes changes in the rate of invading packets. When the rate of invading packets is reduced, the link the attacking packets come from can be deduced.

In the packet marking schemes [1, 3-7], routers along attack path(s) mark packets with partial path information, and the victim extracts path information from the marked packets to reconstruct attack path(s).

ICMP traceback message [8-9] is that when forwarding packets, routers can (with a low probability) send some ICMP traceback messages with some path information to the destination. The destination receives ICMP traceback messages and reconstructs the attack path(s).

Logging-based schemes [10-15] for IP traceback are to log traffic information in the network (such as routers or dedicated storage devices). After attacking events occur, the attacking traffic can be traced back over hop-by-hop upstream deployed routers or deployed ASes.

Hybrid IP traceback [16-19] is designed to make use of advantages of packet marking and logging-based schemes, but introduce their weaknesses.

In overlay network for IP traceback [20], tunnels are made between border routers and a central traceback router. And the central traceback router connects to an intrusion detection system, which checks whether receiving packets are attacking ones or not. If so, the central traceback router knows which border router the attacking packets come from.

In a summary, most approaches for IP traceback require dedicated traceback software or hardware to be deployed at the routers, and are difficult to be deployed incrementally.

B. Logging-based schemes for IP traceback

Existing logging-based schemes can be sub-categorized as three types depending on whether a scheme is packet-level, flow-level or source & destination-level. Thereinto, "packet-level", "flow-level" and "source & destination-level" represent that trace records stored in the network are based on individual packets, flows (5 tuples), and source/destination address pair, respectively. Compared with the first two types, the granularity of the third type is coarser. In the following portion, "router-level" and "AS-level" mean that the reconstructed path consists of a sequent of IP addresses of routers and ASNs, respectively.

1) Packet-level schemes.

[10-12] are router-level. In [10], every SPIE-enhanced routers logs the hash value of every egress packet in Data Generation Agent (DGA) associated with it. [11] finds that SPIE may return misleading results in some special cases. Thus, [11] extends the SPIE, and the authors argue that the proposed extensions can get correct traceback results in the same cases and improve the efficiencies of traceback queries and storage. [12] improves SPIE. Every TOPO-equipped router logs the digest of every ingress packet and its corresponding predecessor identifier (input port number for directly connected link, or source MAC addresses of Ethernet frame for shared bus) locally. When tracing back, the TOPO-equipped router can identify which upstream router the attacking packet comes from.

[13] is AS-level. In [13], border routers of deployed ASes record each ingress and egress packet locally. If a packet is only record as ingress one by a deployed AS, the AS is the victim AS of the packet; if recorded as ingress and egress packet, transit AS; and if only recorded as egress packet, source AS. A sequent of deployed ASes traversed by the packet form the AS-level forwarding path of the packet.

2) flow-level schemes

[14-15] are router-level. In [14], deployed routers extract 5 tuples in the packet header (source address, destination address, source port, destination port, and protocol) and the source MAC Address from every ingress ethernet frame. And if they do not appearing before, their concatenation is stored locally. Different from [14], [15] first aggregates traffic in the time interval through a router to flows (5-tuple) every a fixed time interval, and store these 5 tuples locally. In [14-15], the attacking traffic is traced back according to its flow characteristics (5 tuples).

3) source & destination-level schemes

[15] proposed another router-level scheme, which aggregates traffic in the time interval through a router according to source/destination address pair of ingress packets every a fixed time interval, and store them locally.

It needs to be emphasized, although EasyTrace is flow-level, it is different from any logging-based scheme mentioned above, because EasyTrace is simultaneously AS-level. Furthermore, EasyTrace can confirm the ingress interface(s) of the BGP router(s) through which some attacking flows enter the EasyTrace-enabled AS.

III. IP TRACEBACK ON THE OVERLAY NETWORK

EasyTrace comprises three mechanisms: the mechanism to build an overlay network, the mechanism to sample and log attacking flows, and the traceback mechanism on the overlay network.

A. Assumption and Definition

We identify two assumptions that motivate our design:

(1) Every EasyTrace-enabled AS registers and opens its Autonomous System Number (ASN) and IP address of its Traceback Server (TS) on the website of a certain organization. The registration incentive is that the traceback can be supplied to other ASes, end-users or intrusion detection systems as a charged service. Thus, every TS knows all EasyTrace-enabled ASes and IP addresses of their respective TSes.

(2) Every TS knows which neighboring AS each external interface of its local AS connects to.

Given any two EasyTrace-enabled ASes, *ASi* and *ASj*, if there exists a route from *ASi* to *ASj* without transiting any other EasyTrace-enabled ASes, *ASj* is referred to as a *downstream logical neighbor* of *ASi*. Simultaneously, *ASi* is referred to as an *upstream logical neighbor* of *ASj*. The upstream and downstream logical neighbors are collectively called *logical neighbors*. Given any two ASes, *ASm* and

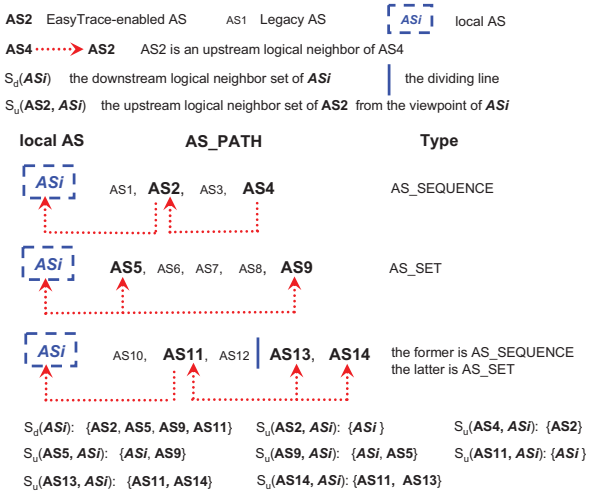

Figure 2. Example of inferring downstream and upstream logical neighbor relations from three types of the typical AS_PATHs in the Historical Route Information Base (HRIB) of *TSi* .

ASp, if *ASm* directly connects to *ASp* through a physical link, *ASm* and *ASp* are referred to as a *physical neighbor* of each other.

B. *Intra-AS Structure*

Fig. 1 shows the intra-AS structure in EasyTrace. The BGP routers enable xFlow function on interfaces to other ASes in order to sample ingress traffic. Additionally, they send sampled traffic information to a Collector which uniformly processes xFlow information in order to serve traceback. Depending on the scale of the AS, there are one or more Collectors in the AS.

Each EasyTrace-enabled AS has a TS in function. The TS learns iBGP routes from iBGP peers of its local AS and computes the best routes. And the TS does not forward traffic or announce any prefix. The TS logs its entire historical best routes into a Historical Route Information Base (HRIB).

The TS is the interface to other TSes, end-users or intrusion detection systems. When a traceback request is presented to the TS, it verifies the authenticity and integrity of the request, queries the Collector of its local AS, and dispatches the traceback request(s) to other appropriate TS(es). When the traceback process is terminated by the TS, the TS replies to the end-users or the intrusion detection systems with the reconstructed attacking path.

C. *Building an AS-Level Overlay Network*

The overlay network plays an important role in the incremental deployment of EasyTrace. The overlay network enables every TS to know who the upstream logical neighbors of its local AS are. Consequently, after a TS affirms that its local AS is in the attacking path by EasyTrace, it will send new traceback request(s) to the TS(es) of upstream logical neighbor(s) of its local AS for the further traceback (two different cases will be described detailed in Ⅲ.E). In the following portion, assume that *ASi* is any one EasyTrace-enabled AS, and *ASj* is any one downstream logical neighbor of *ASi*. We will take *ASi* and *ASj* for example, whose TSes are *TSi* and *TSj*, respectively.

1) *Pre-processing in each Traceback Server (TS)*

The AS_PATH implies the upstream and downstream logical neighbor relations between the EasyTrace-enabled ASes. By searching all AS_PATHs of its HRIB, *TSi* maintains two types of sets: the downstream logical neighbor set of *ASi*, and the upstream logical neighbor set (partial) of each EasyTrace-enabled AS which appears in the AS_PATHs of *TSi*'s HRIB. Note that *TSi* knows all EasyTrace-enabled ASes, and AS_PATH is composed of one or more path segments [21]. Every path segment may be of type AS_SEQUENCE or AS_SET. In general, there are three typical combination forms of path segments for AS_PATH: the first and the second are only one path segment of type AS_SEQUENCE and of type AS_SET, respectively; and the third consists of two path segments: the former is of type AS_SEQUENCE, and the latter is of type AS_SET.

- If the AS_PATH is composed of one path segment of type AS_SEQUENCE, *TSi* will search the AS_PATH from left to right. If the first EasyTrace-enabled AS is found, it is a downstream logical neighbor of *ASi*. Simultaneously, *ASi* is its upstream logical neighbor. And the searching process continues. If the second EasyTrace-enabled AS is found, the first EasyTrace-enabled AS is an upstream logical neighbor of the second, and so on. The searching process continues until the end of the AS_PATH.
- If the AS_PATH consists of one path segment of type AS_SET, all EasyTrace-enabled ASes in the AS_PATH are regarded as the downstream logical neighbors of *ASi*. And each EasyTrace-enabled AS in the AS_PATH regards other EasyTrace-enabled ASes in the AS_PATH and *ASi* as its upstream logical neighbors.
- If the AS_PATH is composed of two path segments: the former is of type AS_SEQUENCE, and the latter is of type AS_SET. The searching process is similar to above two cases.

Fig. 2 shows the example of this searching process. *TSi* extracts the downstream and upstream logical neighbor relations from each AS_PATH in its HRIB. As a result, *TSi* knows the downstream logical neighbor set of *ASi*, $S_d(ASi)$, and the upstream logical neighbor set of each EasyTrace-enabled AS which appears in the AS_PATHs in its HRIB

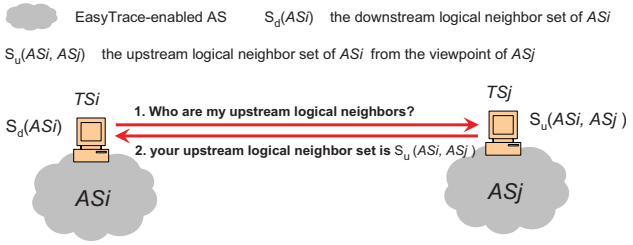

Figure 3. Discovering upstream logical neighbors of *ASi*. ($\forall ASj \in S_d(ASi)$)

from the viewpoint of *ASi*. The pre-processing is the same for any TS.

2) Upstream Logical Neighbor Discovering

According to Ⅲ.C.1, *TSi* maintains the downstream logical neighbor set of *ASi*, $S_d(ASi)$. *TSi* sends a query request to the TS of every member in $S_d(ASi)$, asking which ASes the upstream logical neighbors of *ASi* are. Assuming that *ASj* is any one downstream logical neighbor of *ASi*. *TSj* will receive the query request from *TSi*. According to Ⅲ.C.1, *TSj* maintains the upstream logical neighbor set of *ASi* from the viewpoint of *ASj*, $S_u(ASi, ASj)$. *TSj* will respond to *TSi* with $S_u(ASi, ASj)$. Fig. 3 describes the process that *TSi* talks with *TSj*. When *TSi* receives responses from the TSes of all members in $S_d(ASi)$, *TSi* will get the upstream logical neighbor set of *ASi*, $S_u(ASi)$. That is to say,

$$S_u(ASi) = \bigcup S_u(ASi, ASj) \quad (ASj \in S_d(ASi)). \quad (1)$$

D. Sampling and Logging Attacking Flows

BGP routers send sampled traffic information to the Collector by means of the xFlow (sFlow, NetFlow and IPFIX) function. NetFlow is a network protocol for collecting IP traffic information, which is similar to IPFIX. In the following section, we will only discuss sFlow and NetFlow. Before the Collector uniformly deals with sampled information, two issues must be solved: firstly, timestamps of xFlow export packets are based on the time clocks of different BGP routers, which are hard to be time-synchronized; secondly, the information granularities of sFlow and NetFlow export packets are different. In EasyTrace, we design two mechanisms, the time synchronization mechanism as well as the aggregation mechanism on sFlow packets, to solve the above two issues, respectively.

1) Time synchronization mechanism

xFlow export packets are timestamped by different routers, which are hard to be time-synchronized. In EasyTrace, the timestamps of xFlow export packets are uniformly modified according to the time clock of the Collector by the Collector. Thus, in EasyTrace, traffic information from different BGP routers can be stored in the time order at the Collector.

In the sFlow export packet header, there is one timestamp, called *uptime*, indicating the time when the router sends the export packet to the Collector. In the NetFlow export packet, which contains one NetFlow header and one or more flow records, there are three major types of timestamps [22-23]: *sysUpTime* (*System Uptime*), *first_switched* (*first*) and *last_switched* (*last*). The first timestamp, located in the NetFlow header, indicates the time when the router sends the NetFlow export packet to the Collector. The second and third timestamps, located in the NetFlow flow record, indicate the times when the first packet and the last packet of a flow are observed at a router, respectively. The above three timestamps in the NetFlow export packet are all based on the time clock of the router. Compared to the first timestamp, the relative times of the second and the third timestamps may be computed.

In EasyTrace, when the Collector receives the sFlow (NetFlow) export packet, the *uptime* (*sysUpTime/System Uptime*) value in the packet header is replaced by the time of the Collector when the collector receives the packet. Furthermore, for the NetFlow export packet, *first_switched* (*first*) and *last_switched* (*last*) values in its flow records may be correspondingly modified to the times of the Collector according to the relative times compared to *sysUpTime* (*System Uptime*). Thus, the timestamps of the xFlow export packets are uniformly synchronized with the Collector.

2) Aggregation mechanism on sFlow packets.

Traffic information, sampled and sent to the Collector by xFlow, is different in granularity. The sFlow information is packet-level while the NetFlow information is flow-level. Concretely, after sampling, NetFlow aggregates sampled packet headers to flow-level information and then sends them to the Collector according to a certain configuration, while sFlow, with fewer further processing, sends packet headers and related trajectory information to the Collector.

In EasyTrace, the flow is defined as 6 tuples (*ingress interface index of the router*, *source address*, *destination address*, *source port*, *destination port* and *protocol*). After receiving NetFlow export packets, the Collector can directly obtain flow-level information. But after receiving a series of sFlow export packets which contain some sampled packet headers, the Collector will simulate the processing of NetFlow at routers and aggregate sampled packet headers to flow-level information, benefiting EasyTrace. Furthermore, the Collector will extract the IP address of the sending router from the IP header of the xFlow export packet. Thus, trace records stored at the Collector contain the following 9 properties: *IP address of the router*, *ingress interface index of the router*, *source address*, *destination address*, *source port*, *destination port*, *protocol*, *first* and *last*. The meanings of *first* and *last* are the same as the NetFlow export packet.

E. Traceback on the Overlay Network

In EasyTrace, the attacking flow is traced back over hop-by-hop upstream logical neighbor, every time checking if an upstream EasyTrace-enabled AS may have sampled packets pertaining to the attacking flow.

Before the traceback process begins, the traceback-launched entity *E* (the victim or an intrusion detection system) will identify the 5 tuples *5-tuple* of the attacking flow (*source address*, *destination address*, *source port*, *destination port* and *protocol*). EasyTrace places one constraint on *E*: the attacking flow must be identified in a timely fashion, which stems from the fact that traceback must be initiated before the appropriate trace records are overwritten in the Collector. This time constraint is directly related to the amount of resources dedicated to the storage of trace records.

In a traceback request *req*(*5-tuple*, *P*, *E*), *P* is the partial attacking path that has been successfully

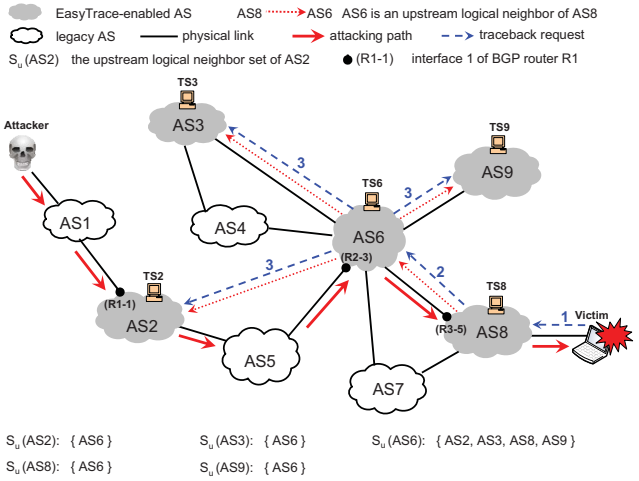

Figure 4. Example of traceback process on the AS-level overlay network by EasyTrace. If AS2, AS6 and AS8 all have sampled the attacking flow, the reconstructed attacking path is [AS1, AS2-R1-1, AS5, AS6-R2-3, AS8-R3-5]. (AS2-R1-1 represents the interface 1 of BGP router R1 in the AS2, others are similar).

reconstructed. If *req*(*5-tuple*, *P*, *E*) is sent by *E*, *P* is null. When any TS *TSi* receives a *req*(*5-tuple*, *P*, *E*) from *E* (or any other TS *TSj*), *TSi* cryptographically verifies its authenticity and integrity.

Upon successful verification, *TSi* executes the local traceback. Concretely, *TSi* will look up trace records in the Collector. Every flow record stored in the Collector of *ASi* has the form of 9 properties as described in Ⅲ.D.2. Thus, if a flow record matches *5-tuple*, the ingress interface *ifindex* of the BGP router *Rp* in *ASi* from which the attacking flow *5-tuple* enters the *ASi* can be identified. Furthermore, according to the 2th assumption in Ⅲ.A, *TSi* knows *ASi* connects to physical neighboring *ASm* through *ifindex* of *Rp* in *ASi*. So *ASm* can be subsequently identified. That is to say, if a flow record in the Collector of *ASi* is hit, *ASi-Rp-ifindex* and *ASm* can successively be proved to be in the attacking path. In the traceback process, depending on whether physical neighbor *ASm* of *ASi* is deployed or not, *ASi* will send different request contents and request numbers for the further traceback.

1) *ASm* is EasyTrace-enabled. *TSi* will only send one new traceback request *new_req*(*5-tuple*, *P'*, *E*) to *TSm* in order to further identify from which ingress interface and BGP router of *ASm* the attacking flow enters *ASm*. Thereinto, *P'* ="*ASi-Rp-ifindex*"+ *P*. When not sampling the attacking flow, *TSm* will tell *TSi*, and *TSi* will terminate the traceback process.

2) *ASm* is **not** EasyTrace-enabled. *TSi* will send a *new_req*(*5-tuple*, *P'*, *E*) to the TS of every member in $S_u(ASi)$ (excluding AS members in *P*). Thereinto, *P'* ="*ASm*, *ASi-Rp-ifindex*"+*P*. When any one of the following three conditions is satisfied, *TSi* will terminate the traceback process: (1) $S_u(ASi)$ is empty. (2) All members in $S_u(ASi)$ are in the *P*; (3) All members in $S_u(ASi)$ have not sampled or forwarded the attacking flow.

After terminating the traceback process, *TSi* will send the traceback result to the traceback-launched entity *E*.

Fig. 4 illustrates the example of a traceback process. Firstly, each EasyTrace-enabled AS knows its upstream logical neighbors by the upstream logical neighbor discovering. When an attacker is attacking the victim, the victim launches the traceback and sends a traceback request to TS8. If AS8 has sampled and forwarded this attacking flow, the interface 5 of BGP router R3 in AS8 is affirmed and then AS6 will be correspondingly identified. And TS8 will send a new traceback request to TS6. If AS6 has sampled and forwarded this attacking flow, the interface 3 of R2 in AS6 is affirmed and then AS5 is correspondingly identified. TS6 will send new traceback requests to the TSes of the upstream logical neighbors of AS6 (excluding AS8), TS2, TS3 and TS9. After looking up their own Collectors, TS3 and TS9 will respond without sampling or forwarding this attacking flow. If AS2 has sampled and forwarded this attacking flow, the interface 1 of R1 in AS2 is affirmed and then AS1 is correspondingly identified. The upstream logical neighbor of AS2 is AS6, but AS6 has been successfully traced back. So the traceback process is terminated at TS2, and TS2 will respond to the victim with the reconstructed path, [AS1, AS2-R1-1, AS5, AS6-R2-3, AS8-R3-5].

IV. EVALUATION

A DDoS attack comprises many attacking flows. In the following section, we will analyze the successfully-traced back probability for any one attacking flow of the DDoS attack. Based on this, we may make the analysis of the DDoS attack.

Assuming that a *m*-packet attacking flow transits *r* EasyTrace-enabled ASes ($r \geq 1$) along one attacking path, and the attacking flow is independently sampled with probabilities p_r, …, p_2, p_1 by r EasyTrace-enabled ASes from the attacker to the victim, respectively. According to the characteristic of traceback process in EasyTrace, the attacking path is traced back hop by hop from the victim to the attacker on the AS-level overlay network. We will analyze the probability P_s that EasyTrace can successfully trace back the *m*-packet attacking flow *h* AS-level hops ($1 \leq h \leq r$, *h* **is sum of EasyTrace-enabled ASes**) from the victim to the attacker on the following two cases.

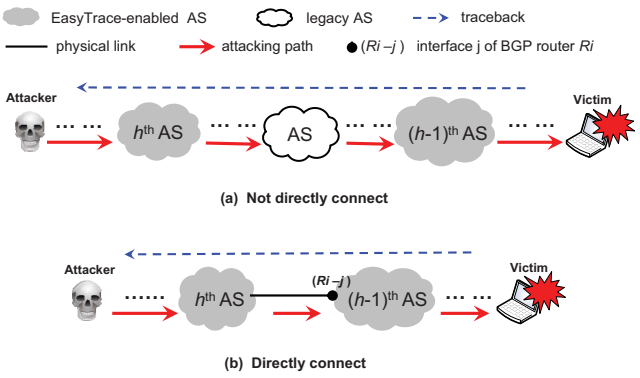

Figure 5. Two types of connecting relations between the $(h\text{-}1)^{th}$ and the h^{th} EasyTrace-enabled AS.

1) The h^{th} and the $(h\text{-}1)^{th}$ EasyTrace-enabled ASes do not directly connect with each other through a physical link, and between them there does not exist any other EasyTrace-enabled AS (Fig. 5.a). In this case, if EasyTrace can successfully trace back the attacking flow h AS-level hops from the victim to the attacker, the h EasyTrace-enabled ASes must all sample the attacking flow. Thus,

$$P_S = \prod_{k=1}^{h} [1-(1-p_k)^m] \qquad (1 \le h \le r). \tag{2}$$

2) The h^{th} EasyTrace-enabled AS is a physical neighbor of the $(h\text{-}1)^{th}$ EasyTrace-enabled AS (Fig. 5.b). If the h-1 EasyTrace-enabled ASes from the victim to the attacker all sample the attacking flow, the interface j of Ri from which the attacking flow enters the $(h\text{-}1)^{th}$ EasyTrace-enabled AS can be identified. Moreover, the h^{th} EasyTrace-enabled AS directly connects to the $(h\text{-}1)^{th}$, so the h^{th} EasyTrace-enabled AS can be affirmed, even if the h^{th} EasyTrace-enabled AS does not sample the attacking flow. In this case,

$$P_S = \prod_{k=1}^{h-1} [1-(1-p_k)^m] \qquad (1 \le h \le r). \tag{3}$$

V. CONCLUSIONS

A sampling-based IP traceback approach is proposed to probabilistically find the origin ASes and the paths of attacking flows. Deployment difficulties are the most challenging for most traceback approaches, but EasyTrace, based on existing protocol and functions (such as BGP, sFlow, NetFlow and IPFIX), is deployable for both intra-AS and inter-AS. In the future, we will optimize storage overhead at the Collector, and make experiments in the real network environment constructed according to the topology of CNGI-CERNET2 [24].

ACKNOWLEDGEMENT

This work was Supported by National Science Foundation of China under Grant 61073172, Program for New Century Excellent Talents in University, Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant 20090002110026,and National Basic Research Program ("973" Program) of China under Grant 2009CB320501.

REFERENCES

[1] S. Savage, D.Wetherall, A. R. Karlin, and T. Anderson, "Practical network support for IP traceback," in Proceeding of ACM SIGCOMM, 2000.

[2] H. Burch and B. Cheswick, "Tracing anonymous packets to their approximate source," In Proceedings of 14th Systems Administration Conference, 2000.

[3] D. X. Song and A. Perrig, "Advanced and Authenticated Marking Schemes for IP Traceback," In Proc. INFOCOM, 2001.

[4] M. Ma, "Tabu marking scheme to speedup IP traceback," Computer Networks, vol. 50, Mar. 2006, pp. 3536-3549.

[5] A. Yaar, A. Perrig, and D. Song, "FIT: Fast Internet Traceback," In Proceedings of IEEE INFOCOM, 2005.

[6] M.T. Goodrich, "Probabilistic packet marking for large-scale IP traceback," IEEE/ACM Transactions on Networking, vol. 16, Feb. 2008, pp. 15-24.

[7] Y. Xiang, WL. Zhou, and MY. Guo, "Flexible Deterministic Packet Marking: An IP Traceback System to Find the Real Source of Attacks," IEEE Transaction on Parallel and Distributed Systems, vol. 20, Apr. 2009, pp. 567-580.

[8] H. C. J. Lee, Vrizlynn L. L. Thing, Y. Xu, and M. Ma, "ICMP traceback with cumulative path, an efficient solution for IP traceback," In Proceedings of 5th International Conference on Information and Communications Security, 2003.

[9] S. F. Wu et al., "On Design and Evaluation of 'Intention-Driven' ICMP Traceback," In Proceedings of 10th Int'l. Conf. Comp. Commun. and Nets., 2001.

[10] A.C. Snoeren, C. Alex, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, B. Schwartz, S. T. Kent and W. T. Strayer, "Single-packet IP traceback," IEEE/ACM Transactions on Networking, vol. 10, Dec. 2008, pp. 721–734.

[11] H. Egon, D. E. P. Jr., M. Glenn, "Extensions to the source path isolation engine for precise and efficient log-based IP traceback," Computers & Security, vol. 29, Jun. 2010, pp. 383-392.

[12] L.Zhang, Y. Guan, "TOPO: A Topology-aware Single Packet Attack Traceback Scheme," Securecomm and Workshops, 2006.

[13] H. Hazeyama, K. Wakasa and T. Takemori, "A Field Trial of Inter-Domain Traceback Operation in Japan," http://tools.ietf.org/html/draft-hazeyama-traceback-field-trial-00, Feb. 2010.

[14] H. Jang, H. Yun and S. Lee, "Attack Flow Traceback," Third International Conference on Convergence and Hybrid Information Technology, 2008.

[15] TH. Lee, WK. Wu, TY. W. Huang, "Scalable packet digesting schemes for IP traceback", IEEE International Conference on Communications, 2004.

[16] B. Duwairi and G. Manimaran, "Novel Hybrid Schemes Employing Packet Marking and Logging for IP Traceback," IEEE Transaction on Parallel and Distributed Systems, vol. 17, May. 2006, pp. 403-418.

[17] C. Gong and K. Sarac, "IP traceback based on packet marking and logging Source," In Proceedings of IEEE International Conference on Communications (ICC), Seoul, Korea, 2005.

[18] C. Gong and K. Sarac, "A more practical approach for single-packet IP traceback using packet logging and marking," IEEE Transaction on Parallel and Distributed Systems, vol. 19, Oct. 2008, pp. 1310-1324.

[19] MH. Sung, J. Xu, J. Li, and L. Li, "Large-Scale IP Traceback in High-Speed Internet: Practical Techniques and Information-Theoretic Foundation," IEEE/ACM Transactions on Networking, vol. 16, Dec. 2008, pp. 1253-66.

[20] R. Stone, "Centertrack: An IP Overlay Network for Tracking DoS Floods," In Proc. 9th USENIX Sec. Symp., 2000.

[21] Y. Rekhter, T. Li and S. Hares, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271, Jan. 2006.

[22] B. Claise, T. Li and S. Hares, "Cisco Systems NetFlow Services Export Version 9", RFC 3954, Oct. 2004.

[23] Cisco Systems, Inc, "NetFlow Services Solutions Guide," http://www.cisco.com/en/US/docs/ios/solutions_docs/netflow/nfwhite.html#wp1030098.

[24] CNGI-CERNET2, "China Next Generation Internet project - China Education and Research Network (CERNET)", http://www.cernet2.edu.cn/index_en.htm.