An Incrementally Deployable Flow-Based Scheme for IP Traceback

Hongcheng Tian and Jun Bi, *Member, IEEE*

***Abstract*—IP traceback can be used to find the origins and paths of attacking traffic. However, so far, most approaches for IP traceback are hard to be deployed in the Internet because of deployment difficulties. In this paper, we present an incrementally deployable approach based on sampled flows for IP traceback (SampleTrace). In SampleTrace, 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. According to Bernoulli's Law of Large Numbers, when a large number of attacking flows are practically traced back in the Internet by SampleTrace, the successfully-traced back relative frequency will approach the successfully-traced back probability.**

***Index Terms*—IP traceback, flow, 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.

IP traceback is to find the origins and attacking paths of malicious traffic. But most IP traceback approaches are difficult to be deployed in the Internet [1], [2], because dedicated software or hardware needs to be deployed at routers, or most methods are difficult to be incrementally deployed.

In this paper, we propose an incrementally deployable IP traceback approach based on sampled flows (SampleTrace). SampleTrace uses existing xFlow (sFlow, NetFlow and IPFIX) function and BGP information to implement traceback, instead of deploying any traceback software or hardware at routers. SampleTrace builds an AS-level overlay network among deployed ASes by the upstream logical neighbor discovering, in order to support the incremental deployment. Furthermore, we design the time synchronization mechanism and the aggregation mechanism for the Collector, in order to uniformly process sampled traffic information sent from different BGP routers and generated from sFlow, NetFlow and IPFIX, respectively. It should be emphasized that, SampleTrace can confirm the ingress interface(s) of the BGP router(s) through which some attacking flows enter the deployed AS. Theoretical analyses show that the probability that an attacking flow is successfully traced back various AS-level hop number quantitatively depends on two factors: independently sampling probability, and the packet number that the flow comprises.

Manuscript received March 4, 2012. The associate editor coordinating the review of this letter and approving it for publication was C. Mitchell.

The authors are with the Network Research Center, Tsinghua University, Beijing, China (e-mail: {tianhc@netarchlab., junbi@}tsinghua.edu.cn).

This work has been supported by National Science Foundation of China under Grant 61073172, Program for New Century Excellent Talents in University, and Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant 20090002110026.

Digital Object Identifier 10.1109/LCOMM.2012.051512.120467

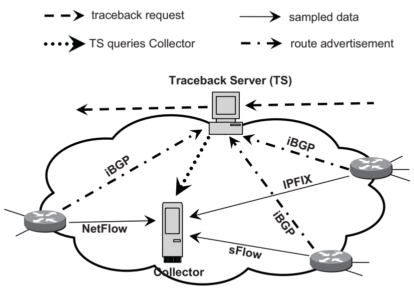

Fig. 1. Intra-AS structure in SampleTrace, consisting of a Traceback Server (TS), one or more Collectors.

SampleTrace has three deployment incentives: 1) A deployed AS can provide the traceback service to other ASes, end-users or intrusion detection systems (IDS) as a charged service; 2) For a deployed stub AS, when users of the stub AS are attacked, SampleTrace can identify which interfaces some attacking traffic enters the stub AS from, and then actions at the interfaces (such as packet filtering or traffic constraint) can be taken to protect its users; 3) If a transit AS can provide more services, such as a traceback service, it is more attractive to potential customer ASes.

II. IP Traceback Based on Flows

A. Assumption and Definition

We identify two assumptions that motivate our design: 1) Every deployed 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 IDS as a charged service. Thus, every TS knows all deployed 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 deployed ASes, *ASi* and *ASj*, if there exists a route from *ASi* to *ASj* without transiting any other deployed ASes, *ASj* is referred to as a *downstream logical neighbor* of *ASi*. Simultaneously, *ASi* is referred to as an *upstream logical neighbor* of *ASj*. Given any two ASes, *ASm* and *ASp*, if *ASm* directly connects to *ASp* through a physical link, *ASm* and *ASp* are referred to as a *physical neighbor* of each other.

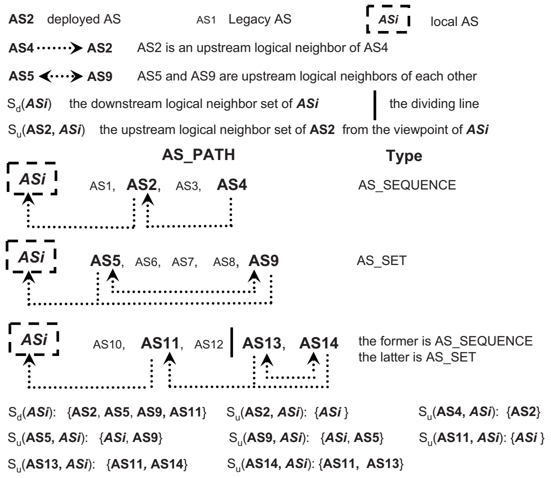

Fig. 2. An example of inferring downstream and upstream logical neighbor relations from three types of the typical AS_PATHs in the Historical Route Information Base (HRIB) of *TSi* (the Traceback Server of *ASi*).

B. Intra-AS Structure

Fig. 1 shows the intra-AS structure in SampleTrace. 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. Depending on the scale of the AS, there are one or more Collectors in the AS.

Each deployed AS has a TS in function (such as, *TSi* of *ASi*). The TS learns iBGP routes from iBGP peers of its local AS and computes the best routes. And the TS does not announce any prefix or forward traffic. 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 IDS.

C. Building an AS-Level Overlay Network

By upstream logical neighbor discovering, every TS knows who the upstream logical neighbors of its local AS are, thus building the overlay network. Consequently, SampleTrace may trace an attacking flow back over hop-by-hop upstream logical neighbor ASes. Assuming that *ASi* is any one deployed AS, and *ASj* is any one downstream logical neighbor of *ASi*, we will take *ASi* and *ASj* for example in the following portion.

1) Pre-processing in each Traceback Server (TS): The AS_PATH implies the upstream and downstream logical neighbor relations between the deployed ASes. By analyzing all AS_PATHs of its HRIB, *TSi* maintains two types of sets: the downstream logical neighbor set of *ASi*, S_d(*ASi*), and the upstream logical neighbor set (partial) of each deployed AS which appears in the AS_PATHs of *TSi*'s HRIB from the viewpoint of *ASi*. Note that *TSi* knows all deployed ASes, and AS_PATH is composed of one or more path segments [3]. Every path segment may be of type AS_SEQUENCE or AS_SET. And there are three typical combination forms for AS_PATH (as mentioned below). Correspondingly, *TSi* will conduct different searching processes.

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

Fig. 2 shows the example of different searching processes.

2) Upstream Logical Neighbor Discovering: According to II.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 who the upstream logical neighbors of *ASi* are. $\forall ASj \in S_d(ASi)$, *TSj* of *ASj* receives the query request from *TSi*. And *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*). 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*).

D. Sampling and Logging Attacking Flows

Since IPFIX is similar to NetFlow, we will only discuss sFlow and NetFlow in the following section. Before the Collector uniformly deals with sampled information, two issues must be solved: 1) timestamps of xFlow export packets are based on the clocks of different BGP routers, which are not easy to be synchronized; 2) sFlow and NetFlow export packets are different in granularity. In SampleTrace, we design the time synchronization mechanism and the aggregation mechanism for the Collector, to solve the above two issues, respectively.

Our objective is below: it can be uniformly stored in the form of trace records at the Collector when and where flows enter the deployed AS from the viewpoint of the Collector.

1) Time synchronization mechanism: In SampleTrace, the timestamps of xFlow export packets are uniformly modified by the Collector and get synchronized to the Collector clock.

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 containing one NetFlow header and one or more flow records, there are three major types of timestamps: *sysUpTime* (*System Uptime*), *first_switched* (*first*) and *last_switched* (*last*), indicating the time when the router sends the NetFlow export packet to the Collector, the times when the first and the last packets of a flow are observed at an interface of a router, respectively. Compared to the first timestamp, the relative times of the second and third timestamps may be computed.

When receiving a sFlow (NetFlow) export packet, the Collector will replace the *uptime* (*sysUpTime/System Uptime*) value in the packet header with the receiving time of the packet. Furthermore, for the NetFlow export packet,

```
// 5-tuple is 5 tuples of the attacking flow
// P and P' are the partial attacking paths which have been reconstructed
// E is the IP address of the traceback-launched entity
// ifindex of Rp means the interface ifindex of BGP router Rp
// result is the traceback result
Let req(5-tuple, P, E) be a traceback request sent by TSj (or E) to TSi;
Let req(5-tuple, P', E) be a new traceback request sent by TSi;
TSi looks up trace records in the Collector of ASi for 5-tuple;
if (a trace record is hit)   then
     it is confirmed that the attacking flow enters ASi from its
            physical neighbor ASm through the ifindex of Rp in ASi;
     respond to TSj (or E) with the HIT message;
     result : = "ASm, ASi-Rp-ifindex," + P ;
     if (ASm is deployed)  then
          P' : = "ASi-Rp-ifindex,"+ P ;
          send req(5-tuple, P', E) to TSm;
          wait the response from TSm;
          if (the respond is the NO HIT message)   then
                respond to E with result ;
     else
          if [(Su(ASi) is empty) or
                    (all members in Su(ASi) are in P)]   then
                respond to E with result ;
          else
                P' : = "ASm, ASi-Rp-ifindex," + P ;
                for each member ASk  in  Su(ASi)
                     if ASk is not in P  then
                            send req(5-tuple, P', E) to TSk;
                wait the responses;
                if (all responds are the NO HIT messages)  then
                     respond to E with result ;
else
     if [req(5-tuple, P, E) is from TSj]  then
          respond to TSj with the NO HIT message;
     if [req(5-tuple, P, E) is from E]  then
          respond to E with the TRACEBACK FAIL message
```

Fig. 3. Traceback algorithm in SampleTrace. We take *TSi* for example. *TSi* executes this algorithm after receiving the traceback request *req*(*5-tuple*, *P*, *E*) from *TSj* (or *E*). Note that (1) *ASj* is a downstream logical neighbor of *ASi*; (2) *ASm* is a physical neighbor of *ASi*; (3) *ASk* is any one upstream logical neighbor of *ASi*. *TSi* sends the **HIT** message or the **NO HIT** message to *TSj*, indicating whether a trace record is hit in the Collector of *ASi* or not. If *TSi* sends the **TRACEBACK FAIL** message to the traceback-launched entity *E* (the victim or the IDS), it shows that the traceback fails.

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 first timestamp after the previous replacement and the pre-computed relative times.

2) Aggregation mechanism: The sFlow information is packet-level while the NetFlow information is flow-level. In SampleTrace, the flow is defined as 6 tuples (*ingress interface index of the sending router*, *src./dst. address*, *src./dst. port* and *protocol*). After receiving a series of sFlow export packets which contain some sampled packet headers, the Collector will simulate NetFlow and aggregate sampled packet headers to flow-level information.

Trace records stored at the Collector contain the following 9 properties: *IP address of the sending router*, *ingress interface index of the sending router*, *src./dst. address*, *src./dst. port*, *protocol*, *first* and *last* (*first* and *last* mean the same as the NetFlow export packet). At the Collector, trace records are grouped according to different sending routers and interfaces, and stored in the time order within each group.

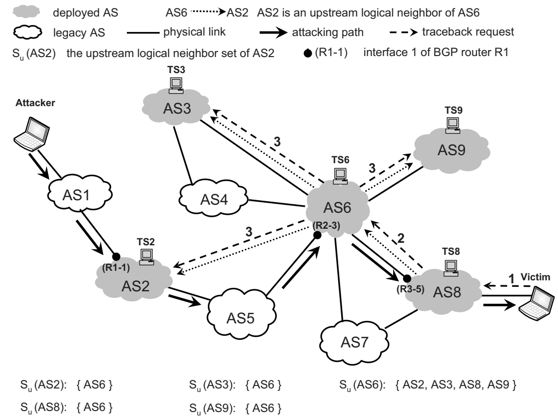

Fig. 4. A traceback example by SampleTrace. Every TS knows the upstream logical neighbors of its local AS by the upstream logical neighbor discovering. Thus, the overlay network is built. When a victim is attacked, the attacking flow will be traced back from the victim to the attacker. The victim will send a traceback request to TS8. If AS8 has sampled the attacking flow, it can be identified that the attacking flow enters AS8 through (R3-5). TS8 knows it is AS6 that connects to AS8 through (R3-5) and AS6 is deployed, so the attacking flow comes from AS6, and TS8 only send one traceback request to TS6. If AS6 has sampled the attacking flow, it can be identified that the attacking flow enters AS6 from AS5 through (R2-3) in AS6. Since TS6 knows AS5 is not deployed, TS6 send one traceback request to the TS of every upstream logical neighbors of AS6 (TS2, TS3, TS9, excluding TS8). Similarly, if AS2 has sampled the attacking flow, AS1 and AS2-R1-1 can be identified (AS2-R1-1 represents the interface 1 of BGP router R1 in AS2). The upstream logical neighbor of AS2 is AS6, but AS6 has been identified. Thus, the traceback process is terminated at TS2, and the reconstructed attacking path is [AS1, AS2-R1-1, AS5, AS6-R2-3, AS8-R3-5]. Maybe, only back part of the attacking path can be reconstructed due to the sampling characteristics.

E. Traceback on the Overlay Network

In SampleTrace, the attacking flow is traced back over hop-by-hop upstream logical neighbor ASes, every time checking if an upstream deployed AS has sampled the attacking flow.

Before the traceback begins, the traceback-launched entity *E*(the victim or the IDS) will identify the 5 tuples *5-tuple* of the attacking flow (*src./dst. address*, *src./dst. port* and *protocol*). In a traceback request *req*(*5-tuple, P, E*), *P* is the partial attacking path that has been successfully reconstructed. 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 traceback algorithm (Fig. 3). Every trace record stored in the Collector of *ASi* has the form of 9 properties as described in II.D.2. Thus, if a trace record matches *5-tuple*, the ingress interface *ifindex* of the BGP router *Rp* in *ASi* from which the attacking flow *5-tuple* enters *ASi* can be identified. Furthermore, according to the 2^{th} assumption in II.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, *ASm* and *ASi-Rp-ifindex* can be proved to be in the attacking path. In the next step, depending on whether physical neighbor *ASm* of *ASi* is deployed or not, *ASi* will send different numbers of new requests *req*(*5-tuple*, P', *E*) for further traceback. Thereinto, P' includes *ASi-Rp-ifindex* and *P*, but excludes the AS which the receiver TS of *req*(*5-tuple*, P', *E*) is attached to.

1) *ASm* is deployed. *TSi* will only send one new traceback request *req*(*5-tuple*, P', *E*) to *TSm* in order to further identify

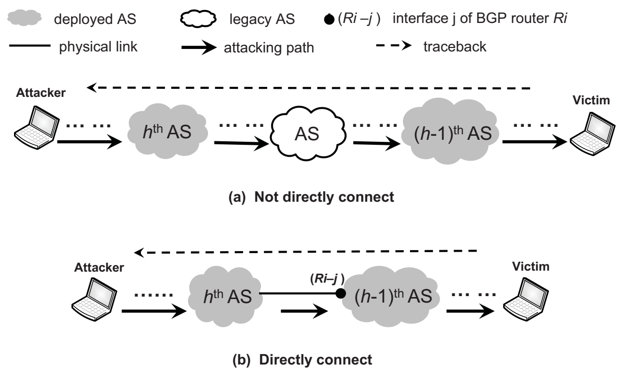

Fig. 5. Two types of connecting relations between the $(h-1)^{th}$ and the h^{th} deployed AS.

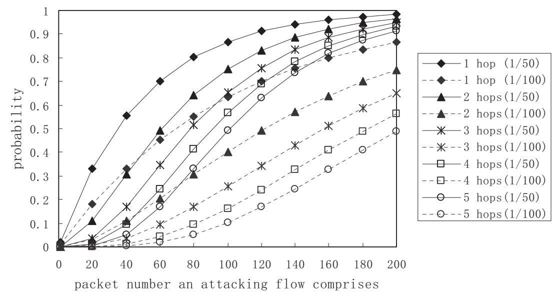

Fig. 6. Probabilities that an attacking flow is successfully traced back various AS-level hop number (the independently sampling probabilities are 1/50 and 1/100, respectively).

from which BGP router and ingress interface of *ASm* the attacking flow enters *ASm*. When not sampling the attacking flow, *TSm* will send **NO HIT** message to *TSi*. And *TSi* will terminate the traceback process.

2) *ASm* is **not** deployed. *TSi* will send a *req*(*5-tuple*, P', *E*) to the TS of every AS member in S_u(*ASi*) (excluding AS members in *P*), because *TSi* cannot confirm which upstream logical neighbor of *ASi* has forwarded the attacking flow to *ASi*. When any one of the following three conditions is satisfied, *TSi* will terminate the traceback process: a) S_u(*ASi*) is empty; b) all members in S_u(*ASi*) are in *P*; c) no member in S_u(*ASi*) has sampled or forwarded the attacking flow.

When terminating the traceback process, *TSi* will send the traceback result *result* to *E*.

Fig. 4 illustrates the example of a traceback process.

III. Evaluation

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

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

$$P_s = \prod_{k=1}^{h} [1 - (1 - p_k)^m] \qquad (1 \leqslant h \leqslant r) \tag{1}$$

For the convenience of the analysis, assume that every deployed AS has the same independently sampling probability. Under two independently sampling probabilities, 1/50 and 1/100, Fig. 6 shows: the successfully-traced back probability decreases as the AS-level hop number desired to be successfully traced back increases, and increases as the independently sampling probability or the packet number the attacking flow comprises increases. When deployed ASes have other uniform independently sampling probabilities, the curvilinear trends are similar to the above.

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

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

When a figure is gotten according to Eq.(2), we will find that it is approximate to Fig. 6 under the same assumptions.

IV. Conclusions

SampleTrace is proposed to probabilistically find the origin ASes and the paths of attacking flows. Deployment difficulties are pretty challenging for most traceback approaches, but SampleTrace, based on existing protocol and functions (such as BGP, sFlow, NetFlow and IPFIX), is deployable for both intra-AS and inter-AS. SampleTrace, different from other existing IP traceback approaches, can provide AS-level traceback service based on flows and logging to the victim or IDS. Furthermore, SampleTrace can identify the ingress BGP routers and related interfaces of deployed ASes some attacking flows enter. We also proposed the deployment incentives of SampleTrace. In the future, we will optimize storage overhead at the Collector, and make experiments in CNGI-CERNET2 [4].

References

[1] S. Yu, *et al.*, "Traceback of DDoS attacks using entropy variations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 3, pp. 412–425, Mar. 2011.
[2] M. D. D. Moreira, *et al.*, "A stateless traceback technique for identifying the origin of attacks from a single packet," *2011 IEEE ICC*.
[3] Y. Rekhter, *et al.*, "A Border Gateway Protocol 4," RFC 4271, Jan. 2006.
[4] CNGI-CERNET2, China Next Generation Internet Project–China Education and Research Network. Available: http://www.cernet2.edu.cn.