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 IP-FIX) 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.

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 TRACBACK 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 one or more Collectors.

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.
AS2 is an upstream logical neighbor of AS4. 

PATH and PATH implies the upstream and downstream logical SET. And there are three typical combination forms for SEQUENCE, and the latter is AS_SEQUENCE. PATH (as mentioned below). Correspondingly, AS5 and AS9 are upstream logical neighbors of each other, the former is AS_SEQUENCE. PATH is just one path segment of type AS_PATH. Consists of two path segments: the former

AS_PATH from switched PATHs in the Historical Route

- Information Base (HRIB) of all AS

- Sampled traffic information to a Collector which uniformly deals with

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, SD(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, SD(ASi). TSi sends a query request to the TS of every member in Sd(ASi), asking who the upstream logical neighbors of ASi are. ∀ASj ∈ Sd(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, Su(ASi, ASj). Su(ASi, ASj) will respond to TSi with Su(ASi, ASj). When TSi receives responses from the TSes of all members in Sd(ASi), TSi will get the upstream logical neighbor set of ASi, Su(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,
switched

3. // E is the IP address of the traceback-launched entity
4. // ifindex of Rp means the interface index of BGP router Rp
5. // result is the traceback result
6. Let req(5-tuple, P, E) be a traceback request sent by TSi (or E) to TSm;
7. Let req(5-tuple, P, E) be a new traceback request sent by TSi;
8. TSi looks up trace records in the Collector of ASi for 5-tuple;

9. 

10. 

11. 

12. 

13. 

14. 

15. 

16. 

17. 

18. 

19. 

20. 

21. 

22. 

23. 

24. 

25. 

26. 

27. 

28. 

29. 

30. 

31. 

32. 

33. 

34. 

35. 

36. 

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.

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 TSi), 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 2nd 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
from which BGP router and ingress interface of ASm the attacking flow enters. 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 $S_n(ASi)$ (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) $S_n(ASi)$ is empty; b) all members in $S_n(ASi)$ are in $P$; c) no member in $S_n(ASi)$ 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 ($r \geq 1$) along one attacking path, and the attacking flow is independently sampled with probabilities $p_1, ..., p_2, p_3$ 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 ($1 \leq h \leq r$), $h$ is sum of deployed ASes from the victim to the attacker on the following two cases.

1) The $h^{th}$ deployed AS 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,

$$P_s = \prod_{k=1}^{h-1} [1 - (1 - p_k)^m] \quad (1 \leq h \leq r) \quad (1)$$

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 $R_i$ 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 $(R_i,j)$, so the $h^{th}$ deployed AS can be affirmed, even if the $(h - 1)^{th}$ deployed AS has not sampled the attacking flow. In this case,

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

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