Passive IP Traceback: Disclosing the Locations of IP Spoofers From Path Backscatter

Guang Yao, Jun Bi, Senior Member, IEEE, and Athanasios V. Vasilakos, Senior Member, IEEE

Abstract—It is long known attackers may use forged source IP address to conceal their real locations. To capture the spoofers, a number of IP traceback mechanisms have been proposed. However, due to the challenges of deployment, there has been not a widely adopted IP traceback solution, at least at the Internet level. As a result, the mist on the locations of spoofers has never been dissipated till now. This paper proposes passive IP traceback (PIT) that bypasses the deployment difficulties of IP traceback techniques. PIT investigates Internet Control Message Protocol error messages (named path backscatter) triggered by spoofing traffic, and tracks the spoofers based on public available information (e.g., topology). In this way, PIT can find the spoofers without any deployment requirement. This paper illustrates the causes, collection, and the statistical results on path backscatter, demonstrates the processes and effectiveness of PIT, and shows the captured locations of spoofers through applying PIT on the path backscatter data set. These results can help further reveal IP spoofing, which has been studied for long but never well understood. Though PIT cannot work in all the spoofing attacks, it may be the most useful mechanism to trace spoofers before an Internet-level traceback system has been deployed in real.

Index Terms—Computer network management, computer network security, denial of service (DoS), IP traceback.

I. INTRODUCTION

IP SPOOFING, which means attackers launching attacks with forged source IP addresses, has been recognized as a serious security problem on the Internet for long [1]. By using addresses that are assigned to others or not assigned at all, attackers can avoid exposing their real locations, or enhance the effect of attacking, or launch reflection based attacks. A number of notorious attacks rely on IP spoofing, including SYN flooding, SMURF, DNS amplification etc. A DNS amplification attack which severely degraded the service of a Top Level Domain (TLD) name server is reported in [2]. Though there has been a popular conventional wisdom that DoS attacks are launched from botnets and spoofing is no longer critical, the report of ARBOR on NANOG 50th meeting shows spoofing is still significant in observed DoS attacks [3]. Indeed, based on the captured backscatter messages from UCSD Network Telescopes, spoofing activities are still frequently observed [4].

To capture the origins of IP spoofing traffic is of great importance. As long as the real locations of spoofers are not disclosed, they cannot be deterred from launching further attacks. Even just approaching the spoofers, for example, determining the ASes or networks they reside in, attackers can be located in a smaller area, and filters can be placed closer to the attacker before attacking traffic get aggregated. The last but not the least, identifying the origins of spoofing traffic can help build a reputation system for ASes, which would be helpful to push the corresponding ISPs to verify IP source address.

B. Motivation

However, to capture the origins of IP spoofing traffic on the Internet is thorny. The research of identifying the origin of spoofing traffic is categorized in IP traceback. To build an IP traceback system on the Internet faces at least two critical challenges. The first one is the cost to adopt a traceback mechanism in the routing system. Existing traceback mechanisms are either not widely supported by current commodity routers (packet marking [5]), or will introduce considerable overhead to the routers (Internet Control Message Protocol (ICMP) generation [6], packet logging [7]), especially in high-performance networks. The second one is the difficulty to make Internet service providers (ISPs) collaborate. Since the spoofers could spread over every corner of the world, a single ISP to deploy its own traceback system is almost meaningless. However, ISPs, which are commercial entities with competitive relationships, are generally lack of explicit economic incentive to help clients of the others to trace attacker in their managed ASes. Since the deployment of traceback mechanisms is not of clear gains but apparently high overhead, to the best knowledge of authors, there has been no deployed Internet-scale IP traceback system till now. As a result, despite that there are a lot of IP traceback mechanisms proposed and a large number of spoofing activities observed, the real locations of spoofers still remain a mystery.

Given the difficulties of the IP traceback mechanisms deployment, we are considering another direction:
tracking the spoofers without deploying any additional mechanism. In another word, we try to disclose the location of spoofers from the traces generated by existing widely adopted functions on commodity routers when spoofing attacks happen.

C. Our Work

Instead of proposing another IP traceback mechanism with improved tracking capability, we propose a novel solution, named Passive IP Traceback (PIT), to bypass the challenges in deployment. Routers may fail to forward an IP spoofing packet due to various reasons, e.g., TTL exceeding. In such cases, the routers may generate an ICMP error message (named \textit{path backscatter}) and send the message to the spoofed source address. Because the routers can be close to the spoofers, the path backscatter messages may potentially disclose the locations of the spoofers. PIT exploits these path backscatter messages to find the location of the spoofers. With the locations of the spoofers known, the victim can seek help from the corresponding ISP to filter out the attacking packets, or take other counterattacks. PIT is especially useful for the victims in reflection based spoofing attacks, e.g., DNS amplification attacks. The victims can find the locations of the spoofers directly from the attacking traffic.

In this article, at first we illustrate the generation, types, collection, and the security issues of path backscatter messages in section III. Then in section IV, we present PIT, which tracks the location of the spoofers based on path backscatter messages together with the topology and routing information. We discuss how to apply PIT when both topology and routing are known, or only topology is known, or neither are known respectively. We also present two effective algorithms to apply PIT in large scale networks. In the following section, at first we show the statistical results on path backscatter messages. Then we evaluate the two key mechanisms of PIT which work without routing information. At last, we give the tracking result when applying PIT on the path backscatter message dataset: a number of ASes in which spoofers are found.

Our work has the following contributions:

1) This is the first article known which deeply investigates path backscatter messages. These messages are valuable to help understand spoofing activities. Though Moore et al. [8] has exploited backscatter messages, which are generated by the targets of spoofing messages, to study Denial of Services (DoS), path backscatter messages, which are sent by intermediate devices rather than the targets, have not been used in traceback.

2) A practical and effective IP traceback solution based on path backscatter messages, i.e., PIT, is proposed. PIT bypasses the deployment difficulties of existing IP traceback mechanisms and actually is already in force. Though given the limitation that path backscatter messages are not generated with stable possibility, PIT cannot work in all the attacks, but it does work in a number of spoofing activities. At least it may be the most useful traceback mechanism before an AS-level traceback system has been deployed in real.

3) Through applying PIT on the path backscatter dataset, a number of locations of spoofers are captured and presented. Though this is not a complete list, it is the first known list disclosing the locations of spoofers.

II. RELATED WORK

Though PIT is used to perform IP traceback, it is very different from existing IP traceback mechanisms. PIT is inspired by a number of IP spoofing observation activities. Thus, the related work is composed by two parts. The first briefly introduces existing IP traceback mechanisms, and the second introduces the IP spoofing observation activities.

A. IP Traceback

IP traceback techniques are designed to disclose the real origin of IP traffic or track the path. Existing IP traceback approaches can be classified into five main categories: packet marking, ICMP traceback, logging on the router, link testing, overlay, and hybrid tracing.

Packet marking methods require routers modify the header of the packet to contain the information of the router and forwarding decision. Hence the receiver of the packet can then reconstruct the path of a packet (or an attacking flow) from the received packets. There are two classes of packet marking schemes: probabilistic packet marking [5], [9]–[14] and deterministic packet marking [15]–[18]. Packet marking methods are generally considered to be lightweight because they do not cost storage resource on routers and the link bandwidth resource. However, packet marking is not a widely supported function on routers; thus, it is difficult to enable packet marking traceback in the network.

Different from packet marking methods, ICMP traceback [6], [19], [20] generates addition ICMP messages to a collector or the destination. The ICMP messages can be used to reconstruct the attacking path. For example, if iTrace [6] is enabled, routers generate ICMP samples to destinations with a certain probability. The shortcoming of ICMP traceback is considerable additional traffic will be generated to consume the already stressed bandwidth resource. Moreover, when the attack is against the bandwidth of the victim, the increased traffic will make the attack more serious. ICMP generation can be performed by the processor, but significant overhead will be introduced to the processor.

Attacking path can be reconstructed from log on the router when router makes a record on the packets forwarded [7]. Bloom filter is used to reduce the number of bits to store a packet. Nevertheless, to achieve a low enough collision probability in current high-speed networks, the storage cost is still too large for commodity routers.

Link testing is an approach which determines the upstream of attacking traffic hop-by-hop while the attack is in progress. A controlled flooding mechanism based on performing UDP Chargen request flooding iteratively on the victim rooted tree to see the effects on attacking traffic is proposed in [21]. Because of the huge scale of the Internet, this approach is hard to perform at the Internet level.
CenterTrack [22] proposes offloading the suspect traffic from edge routers to special tracking routers through a overlay network. Though such a mechanism can reduce the requirement on edge routers, the management of the tunnels and the overlay network will be significantly increase the network management overhead. Ref. [23] proposes building an AS-level overlay to trace spoofers. It is found if hundreds of ASes can join the overlay network, the spoofers can be accurately located. However, the challenge in practice is how to make the ASes cooperate. The intra-domain version of this work [24] can avoid this problem, but it is necessary to update routers to adopt modification on OSPF.

The above mechanisms can be combined to achieve better tracing capacity and/or reduce the cost. There are a number of hybrid mechanisms employ both packet marking and logging [25]–[28]. Though the overhead on routers can be reduced, they require the routers to support both mechanisms; thus the barrier to adopt them is higher than adopting a single mechanism.

Though there have been a large number of promising trace-back mechanisms, there is still a long way to get the proposed mechanisms widely deployed, especially at the Internet level. Currently, there is still lack of a ready mechanism to track the spoofers.

B. IP Spoofing Observation

Network telescope [4] is a fundamental technique for passive observation of spoofing activities on the Internet. Network telescope captures non-solicited messages, which are mainly generated by victim attacked by traffic with source prefix set in the scope owned by the telescope. Then, it can be determined a part of nodes which are attacked by spoofing traffic. Currently, the largest scale telescope is the CAIDA UCSD telescope, which owns 1/256 of all the IP addresses and is mainly used to observe DDoS activities and worms. Moore et al. [8] presented a technique named “backscatter analysis” which infers characteristics of DoS attacks based on traces collected by the network telescope. Though ICMP error messages are mentioned in the paper, it does not further investigate these messages to trace spoofers. CAIDA provides publicly accessible data. The main analysis and experimental work of this article are performed on the data supplied by CAIDA.

The MIT Spoofer Project [29] tries to disclose which networks are able to launch spoofing based attacks. Volunteer participants install a client that tests the spoofing ability of their hosts and networks. The statistic result shows 6700 ASs out of 30205 do not filter spoofing.

A recent report from Arbor network based on more than 5000 attacks shows an intriguing result [3]. Unrealistic per IP traffic of 4Gbps is observed in 10% attacks, and significant rate of TCP connections are launched from just a few validated hosts. Though this is not direct evidence of spoofing, it suggests spoofing may be used in such attacks.

In our previous work [30], we presented a preliminary statistical result on path backscatter messages and discussed it is possible to trace spoofers based on the messages. However, the generation and collection of path backscatter messages are not well investigated, and the traceback mechanisms are not designed. In this article, we make a thorough analysis on path backscatter messages, present the traceback mechanisms and give the traceback results.

It can be found existing observations are performed sideways, there is no work has disclosed the locations of spoofers.

III. PATH BACKSCATTER

A. Overview

Not all the packets reach their destinations. A network device may fail to forward a packet due to various reasons. Under certain conditions, it may generate an ICMP error message, i.e., path backscatter messages. The path backscatter messages will be sent to the source IP address indicated in the original packet. If the source address is forged, the messages will be sent to the node who actually owns the address. This means the victims of reflection based attacks, and the hosts whose addresses are used by spoofers, are possibly to collect such messages. This scenario is illustrated in Fig. 1.

As specified by RFC792 [31], the format of the path backscatter messages, is illustrated in Fig. 2. Each message contains the source address of the reflecting device, and the IP header of the original packet. Thus, from each path backscatter, we can get 1) the IP address of the reflecting device which is on the path from the attacker to the destination of the spoofing packet; 2) the IP address of the original destination of the spoofing packet. The original IP header also contains other valuable information, e.g., the remaining TTL of the spoofing packet. Note that due to some network devices may perform address rewrite (e.g., NAT), the original source address and the destination address may be different.
B. Classes and Causes of Path Backscatter

Path backscatter messages can be triggered for various reasons. Based on RFC792, there can be totally 5 types of path backscatter messages, as listed in the following sections. There are a number of *codes* associated with each *type*. The combination of *type* and *code* specifies the cause that the router decides to send the ICMP message. We name the combination of *type* and *code* by *class*. We use the names defined in [32] to denote the *classes* of path backscatter messages. In the path backscatter dataset from CAIDA [4], totally 23 classes of path backscatter messages are found, 11 of them are listed in Table I. Messages belonging to the other 12 types are very rare. We do not find all the possible classes.

We try to explain the causes of the classes of path backscatter messages listed in Table I based on analyzing the dataset. Especially, we try to make out the reasons that they are generated near the spoofers. However, although we have tried our best to explore the possible reasons, considering the sophistication of attacks and the complexity of networks, we do not claim we found all the (or even the main) reasons for the generation of the messages. It should be noted that in general a majority of path backscatter messages are generated near the victim. However, considering the huge number of spoofing messages, if only a small ratio of them trigger path backscatter messages near thespofer, the total path backscatter dataset will be valuable. Even for the path backscatter messages generated far away from the spoofers, their generation locations are at least closer to the spoofers than the victims. Thus, they can be used in the first step of traceback.

1) Time Exceeded: *TIMXCEED_INTRANS* messages are triggered by packets with zero TTL value. Such messages are the most common path backscatter messages. Though the attackers can set the initial TTL value to be large enough to avoid triggering such messages, they may intentionally send packets with small initial TTL values, which trigger routers on the path to generate TTL Exceeding messages to consume the processor resource of the router. In general such attacks target the routers rather than hosts. We also find the attack against a host and the attack against the nearby routers of the target host can be combined. We think the attacker may want to degrade the forwarding performance of the routers near the target host, and then less aggregated spoofing traffic are require to prevent legitimate traffic from reaching the host. Besides, to determine the correct initial TTL value to make sure the TTL exceeding events happen on the targeted router, the attacking hosts should perform some traces. The traces using real address can be cloaked with a number of traces using forged addresses to avoid tracking. This could be the reason that we found a number of *TIMXCEED_INTRANS* messages from cascading routers in the dataset.

2) Destination Unreachable: *UNREACH_FILTER_PROHIB*, *UNREACH_NET_PROHIB* and *UNREACH_HOST_PROHIB* messages are mainly triggered by filtering mechanisms deployed between the spoofing origin and the victim, e.g., Access Control List (ACL). A result of the MIT Spoof project shows 80% filters are deployed one IP hop from the source, and over 95% of blocked packets are filtered at the source AS. Thus, such messages can be from the gateways near the spoofers. It should be noted that at least part of the spoofing traffic from the spoofers has been filtered. Considering the filtering granularity may be coarse, the remaining spoofing messages can still reach the victims. Thus, traceback in such a scenario is still valuable.

UNREACH_HOST and UNREACH_NET messages are generated if there is no route to the destination. Such messages are mostly triggered by attacking traffic launched against a private or unallocated address prefix. Whenever a spoofer sends packets to a private address, if the spoofers is attached to a public network or the victim address is not in the same private network of the spoofer, such ICMP messages will be generated when the spoofing packets arrive at the DFZ (Default-free Zone). We find a large number of such messages whose original destination is a private address. Such messages may be triggered by attacks against hosts behind NAT or in VPN.

UNREACH_NEEDFRAG messages are generated if the size of the attacking packets are larger than the MTU of a hop on the path, but the Don’t Fragment flag is set. Such messages may be generated due to attacks against the routers. Besides, we think such messages can be triggered occasionally. Attackers use large packet to consume the bandwidth of the target. Due to forged addresses are used, the attackers cannot get the ICMP message and are unaware of that the attacking packets are dropped on path.

3) Source Quench: *SOURCEQUENCH* messages are generated when the router has no buffer to queue the original packet. It can be resulted from the aggregated attacking traffic is too large to be forwarded by the router. In general such messages are generated near the victim. However, if there are a large number of attackers in the same network/AS, it is possible to trigger such messages on the gateways near the attackers.

4) Redirect: *REDIRECT_HOST* and *REDIRECT_NET* messages are generated if the spoofing origin has two or more gateways and a gateway, G1, finds the spoofing packet should be sent to another gateway, G2, as this is the shortest path. As multi-homed networks become common, such messages may be generated with higher probability. Because this message is generated by gateways near the spoofing origin, it is particularly helpful to find the location of the origin.

<table>
<thead>
<tr>
<th>Type</th>
<th>Class</th>
<th>Table I</th>
<th>Path Backscatter Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Exceeded</td>
<td>TIMXCEED_INTRANS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination Unreachable</td>
<td>UNREACH_FILTER_PROHIB, UNREACH_NET_PROHIB, UNREACH_HOST_PROHIB, UNREACH_HOST, UNREACH_NET, UNREACH_NEEDFRAG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source Quench</td>
<td>SOURCEQUENCH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Redirect</td>
<td>REDIRECT_HOST, REDIRECT_NET</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter Problem</td>
<td>PARAMPROB</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
As specified in RFC792, G1 should check the address of the packet and G2 are in the same network. However, the dataset is collected by a network telescope, and apparently any G2 and the address of a network telescope must not in the same network. It may be due to misconfiguration or implementations inconsistent with the standard.

5) Parameter Problem: PARAMPROB messages are generated if the router finds a problem with the header parameters in the original packet. Such messages are rare in the dataset. Possibly they are triggered by malformed attacking packets or just some type of attack.

C. Collection of Path Backscatter Messages

Though path backscatter can happen in any spoofing based attacks, it is not always possible to collect the path backscatter messages, as they are sent to the spoofed addresses. We classify spoofing based attacks into four categories, and discuss whether path backscatter messages can be collected in each category of attacks.

1) Multiple Sources, Single Destination: In such attacks, the source address of spoofing packets is chosen from a set of candidate addresses. Particularly, this set contains all the addresses. Such attacks are named random spoofing. Random spoofing is typically used to deplete the resource of the target, e.g., SYN flooding.

Network Telescopes (or darknets) can be used to capture path backscatter messages in random spoofing attacks. As illustrated in Fig. 3, in random spoofing attacks, the path backscatter messages are sent to the randomly spoofed addresses. Because the addresses owned by network telescopes can be used in random spoofing attacks, the network telescopes are possibly able to capture part of the path backscatter messages. Potentially, volunteers can make use of packet capturing tools to collect non-solicited messages, and then they can help in traceback rather than completely relying on the network telescopes. However, this topic is beyond the scope of this work.

2) Single Source, Multiple Destinations: In such attacks, all the spoofing packets have the same source IP address. The packets are sent to different destinations. Such packets are typically used to launch reflection attacks.

Reflection attacks, e.g., DNS amplification, are the most prevalent IP spoofing attacks in recent years.

The victim in a reflection attack is the host who owns the spoofed address. The victim itself is able to capture all the path backscatter messages in reflection attacks. As illustrated in Fig. 4, because all the spoofing packets are sent the address of the victim, all the path backscatter messages will be sent to the victim. Then the victim can get the path backscatter messages through checking if it has sent messages to the original destination IP address field in received ICMP messages.

3) Multiple Sources, Multiple Destinations: Spoofing attacks can be launched against multiple destination IP addresses belonging to the same website or service provider (e.g., cloud). Generally, such attacks can be regarded as the combination of multiple attacks belonging to the above two types.

4) Single Source, Single Destination: Such attacks are often used to hijack or break a session between the source and the destination, e.g., Man-in-the-Middle(MitM), TCP hijack, replay attack. The spoofed origin is able to capture the path backscatter messages. However, because the spoofing packets are generally scarce compared with the other types of attacks, path backscatter messages could be quite rare. On the other hand, because the attacker and the spoofed origin often reside in the same network, it is possible to track the attacker more efficiently than using path backscatter.

In summary, path backscatter messages can be effectively collected in random spoofing attacks, reflection attacks and their combinations, which cover the majority of IP spoofing attacks.

D. Security Issues With Path Backscatter Messages

It should be noted it is almost as easy to fabricate a path backscatter message as to generate a spoofing data packet. Thus, the collector should filter out the forged packet backscatter messages to avoid false positive.

For reflection attack, the victim can get the valid hop count from the routers to itself through tracing or passive learning. Then the mapping from router to hop count can be used to filter out a large part of spoofing packets based on the mechanism proposed in [33]. The attacker must get the correct hop count from each router to the victim to bypass such a filtering mechanism. However, it is difficult and costly for the attacker
to achieve this information, as it cannot get the hop count between the victim and the router from tracing directly. Hop count based filtering can not discard all the spoofing messages, but anyway it makes the spoofing of path backscatter message harder. Moreover, to bypass such filtering, the spoofer has to send some trace or trial messages, and such messages may expose their locations and objectives. Another strategy of the attacker is sending forged path backscatter messages with all the possible TTL values, but the victim can check whether there are path backscatter messages from a node but with various TTL values and/or hop counts to identify such attacks.

For path backscatter messages captured by network telescope in random spoofing, hop count based filtering can also be used by the network telescope itself. However, a third-party who is performing tracing does not know the hop count from each router to the network telescope. In this article, we make use of clustering based mechanism to filter out forged path backscatter messages. We extract all the prefixes from the BGP dataset. For backscatter messages from each prefix,

1) We divided the whole dataset into 1-hour slices. The routing and address assignment are dynamic on the Internet. We chose one hour as the time interval in order to make a trade-off between getting enough data and mitigating the effects of network changes. Note that because network telescopes collect all the non-solicited messages, the dataset contains all the messages, in which only a small portion are path backscatter messages.

2) We inferred the addresses of NAT from each slice. First, we inferred the initial TTL value and hop count of each message (not only path backscatter messages) based on the mechanism proposed in [33]. If an address has multiple initial TTL values but approximate hop counts, the address is considered as a NAT address.

3) For each address other than NAT addresses, we got the mode of hop count value. We filtered out the path backscatter messages whose hop count is deviated from the mode more than 1. We didn’t use exact match for taking the network changes into account. For NAT addresses, we got multiple modes of hop count, and filtered messages whose hop count is deviated from the nearest mode more than 1.

The rationale of this mechanism is the address space of network telescope is hidden. Thus, it will not be targeted, and the received forged path backscatter messages are rare. On the other hand, we make use backscatter messages from hosts together with path backscatter messages. Thus, the dataset is large and the messages are from almost every corner of the Internet. Then the majority of learned hop count should be valid, avoiding pollution from forged path backscatter messages. Although forged path backscatter messages may be of matched hop count accidentally, the possibility will be quite low. To effectively pollute the captured dataset, an attacker will have to send messages with all the possible TTL values to every corner of the Internet. This requires tremendous effort, but the forged messages can still be effectively identified through checking whether messages from a node are with wide-ranged hop counts.

A misunderstanding about the packet backscatter messages is that the normal ICMP error messages, e.g., Time-exceeded messages generated in traceroute, may be regarded as triggered by spoofing packets. However, hosts are using the real source IP address in normal behaviors, and the normal ICMP error messages will go to the hosts themselves rather than the collectors. Thus, innocent hosts triggering normal ICMP error messages will not be regarded as spoofer.

IV. PIT: TRACKING BASED ON PATH BACKSCATTER

We name the IP traceback solution based on exploiting path backscatter messages by Passive IP Traceback (PIT). PIT is actually composed by a set of mechanisms. The basic mechanism, which is based on topology and routing information, is illustrated in section IV-A. However, generally the routing information is hard to achieve. The mechanisms work in case the routing information in unknown are specified in section IV-B. In very special cases, it is possible to track the spoofer without topology and routing information. The mechanism for these cases is discussed in section IV-C.

A. Basic Tracking Mechanism

Whenever a path backscatter message whose source is router $r$ (named reflector) and the original destination is $od$ is captured, the most direct inference is that the packet from attacker to $od$ should bypass $r$. We use a very simple mechanism in spoofing origin tracking.

The network is abstracted as a graph $G(V, E)$, where $V$ is the set of all the network nodes and $E$ is the set of all the links. A network node can be a router or an AS, depending on the tracking scenario. From each path backscatter message, the node $r, r \in V$ which generates the packet and the original destination $od, od \in V$ of the spoofing packet can be got. Denote the location of the spoofer, i.e., the nearest router or the origin AS, by $a, a \in V$.

We make use of path information to help track the location of the spoofer. Use $path(v, u)$ to denote the sequence of nodes on one of the path from $v$ to $u$, and use $PATH(v, u)$ to denote the set of all the paths from $v$ to $u$. Use $\phi(r, od)$ to denote the set of nodes from each of which a packet to $od$ can bypass $r$, i.e.,

$$\phi(r, od) = \{v | r \in path(v, od), path(v, od) \in PATH(v, od)\}.$$

$\phi(r, od)$ actually determines the minimal set which must contain the spoofer. We name the result set of $\phi(r, od)$ by suspect set. As illustrated in Fig. 5, if all the paths are loop-free, the suspect set determined by the path backscatter message is $\{Attacker, Router\}$. If the topology and routes of the network are known, this mechanism can be used to effectively determine the suspect set. For example, an ISP can make this model to locate spoofer in its managed network. However, for most cases, the one who performs tracing does not know the routing choices of the other networks, which are non-public information. Moreover, the topologies of most of the ASes are unknown to the public. In the following sections, we discuss how to track without routing information in section IV-B, and how to track with neither topology nor routing information in section IV-C.
B. Tracking Without Routing Information

It is possible to get the topology of the network in some traceback scenarios. For example, the router-level topology can be got from traceroute, and the AS-level topology can be inferred from the BGP data and supplementary means. Besides, a number of ASes make public their topologies [34]. However, the routes of a network are always treated as business secret and are non-public. In this section, we discuss how to perform PIT if topology is known but the detailed routing is unknown.

It should be noted that if the routing has not constraint, packets from any node \( v \in V \) to \( od \) can bypass any intermediate node \( r \). Then the tracking is meaningless. Fortunately, it is not the case in real networks. We make use of two assumptions on the routing respectively:

1) **Loop-Free Assumption:** This assumption states there is no loop in the paths. This assumption always holds unless misconfiguration or the routing has not converged.

2) **Valley-Free Assumption:** This assumption states there should be no valley in the AS level paths [35]. Though the increased complexity of AS relationship has reduced the universality of this assumption, it is still the most common model of AS level routing.

In the following subsections, we discuss how to perform PIT based on each of the assumption respectively.

1) **Tracking on Loop-Free Assumption:** Based on the loop-free assumption, a vertex \( v \) is in the \( \varphi(r, od) \) if and only if there is at least one loop-free path from \( v \) to \( od \) passing \( r \). Denote a loop-free path from \( v \) to \( u \) by \( l_{fpath}(v, u) \), which is a sequence of verticals along the path. Then the suspect set is

\[
\varphi(r, od) = \{ v | \exists l_{fpath}(v, od), r \in l_{fpath}(v, od) \}.
\]

To find all the satisfying verticals through enumerating is almost impossible for large-scale networks. We designed an algorithm specified in Fig. 6. This algorithm first finds a shortest path from \( r \) to \( od \). From the second vertex along the path, it checks if the removal of the vertex can break \( r \) and \( od \). Whenever such a vertex \( c \) is found, removing the vertex from \( G \), and the set containing all the verticals which are still connected with \( r \) is just the suspect set.

The following theorem can be proofed to illustrate the correctness of the algorithm. The proof of this theorem is placed at the Appendix A.

**Theorem 1:** From the second vertex along \( path(r, od) \), remove the first articulation point \( c \) whose removal will break \( r \) and \( od \). Denote the subgraph containing \( r \) by \( SG(r) \). If and only if \( v \) is in \( SG(r) \), there exists a loop-free path from \( v \) to \( od \) containing \( r \).

Apparantly, to determine a suspet set whose size is no larger than \( N \) requires the vertex number connected with \( r \) is no more than \( N \) in \( G - CutEdge(r, od) \). Especially, if the size of suspect set is 1, the degree of \( r \) must be one, and \( od \) must not be \( r \).

2) **Tracking on Valley-Free Assumption:** Based on the valley-free assumption, a vertex \( v \) is in the \( \varphi(r, od) \) if and only if there is at least one valley-free path from \( v \) to \( od \) passing \( r \). Denote a valley-free path from \( v \) to \( u \) by \( v_{fpath}(v, u) \), which is a sequence of verticals along the path. Then the suspect set is

\[
\varphi(r, od) = \{ v | \exists v_{fpath}(v, od), r \in v_{fpath}(v, od) \}.
\]

The valley-free assumption can be only used in AS-level topology. Considering the scale of AS-level Internet topology, for a path backscatter message \( (r, od) \), it is very costly to find all the ASes that has a valley-free path to \( od \) through \( r \). At first we introduce the concept of customer cone [36], which means “AS A, plus A’s customers, plus its customers’ customers, and so on”. The customer cone of AS \( v \) is denoted by \( Cone(v) \).

Then we can proof the following theorem:

**Theorem 2:** When \( od \notin Cone(r) \), if and only if \( v \in Cone(r) \), there is a valley-free path from \( v \) to \( od \) passing \( r \).
Fig. 7. The algorithm to determine suspect set based on valley-free assumption.

```
1: function GetSuspectSet_ValleyFree(G, r, od)
2:     if od ∈ Cone(r) then
3:         return G.nodes()
4:     else
5:         return Cone(r)
6:     end if
7: end function
```

Fig. 8. The suspect set determined by a path backscatter message with valley-free assumption.

The proof of this theorem is placed at Appendix B. Based on this theorem, when \( od \notin Cone(r) \), the suspect set is just \( Cone(r) \). When \( od \in Cone(r) \), because any valley-free path followed by a downhill path is still a valley-free path, the suspect set is the whole node set (Note that loop-free is not considered here). Thus, the algorithm is as specified in Fig. 7. Fig. 8 illustrate the suspect set tracked based on the valley-free assumption. To determine a suspect set whose size is not larger than \( N \) requires the customer cone size of \( r \) is no larger than \( N \). Especially, if the size of suspect set is 1, the \( r \) should be a stub AS.

C. Tracking Without Topology and Routing Knowledge

The above tracking mechanisms actually have two limitations. The first is the network topology and mapping from addresses of \( r \) and \( od \) must be known. The second is the tracking is actually performed based on loose assumptions on paths. Thus, only when path backscatter messages are from very special vertex, i.e., stub AS, the spoofer can be accurately located. In this section, we discuss how to break these limitations through using other information contained in path backscatter messages.

We found there are three special types of path backscatter messages which are more useful for tracing spoofer:

1) The path backscatter messages whose original hop count is 0 or 1. Such messages are generated 1 or 2 hops from the spoofer. Very possibly they are from the gateway of the spoofer.

2) The path backscatter messages whose type is ‘Redirect’. Such messages must be from a gateway of the spoofer.

3) The path backscatter messages whose original destination is a private address or unallocated address. Such messages are typically generated by the first DFZ router on the path from the spoofer to the original destination, e.g., the egress router of the AS in which the spoofer resides.

Though such path backscatter messages are generated in very special cases, they are not rare. Especially, there are a large number of path backscatter messages whose original destination is a private address.

V. Evaluation

PIT is very different from any existing traceback mechanism. The main difference is the generation of path backscatter message is not of a certain probability. Thus, we separate the evaluation into 3 parts: the first is the statistical results on path backscatter messages; the second is the evaluation on the traceback mechanisms presented in section IV-B without considering uncertainty of path backscatter generation, since effectiveness of the mechanisms is actually determined by the structure features of the networks; the last is the result of performing the traceback mechanisms on the path backscatter message dataset. The datasets used in the evaluation are listed in Table II. The path backscatter messages are extracted from the newest backscatter dataset from CAIDA, which contains messages collected in 37 days across 5 months in 2008.

A. Statistical Results on Path Backscatter Messages

1) Overview: Though the generation of path backscatter is of no inevitability, the total volume of path backscatter is quite notable. In the backscatter dataset, we found totally 175,695,985 path backscatter messages, involving 283,689 reflector IP addresses and 3,205,540 original destination IP addresses. The number of special messages mentioned in Section IV-C is 1,199,039. The number (reflector IP, original destination IP) tuples (named path backscatter IP tuples, or simply IP tuples) is 3,721,827.

At AS level, there are 32526 (reflector AS, original destination AS) tuples (named path backscatter AS tuples, or simply AS tuples), involving 7042 reflector ASes and 9198 original destination ASes. 26376 path backscatter AS tuples have different reflector AS and original destination AS. There are 4327 path backscatter AS tuples whose source IP or original destination IP is private or unallocated, involving 4,159,844 path backscatter messages and 319,570 path backscatter IP tuples.
2) Distribution of Related IP Addresses and ASes: We plot the path backscatter IP tuples (aggregated in/16 granularity) in Fig. 9. It can be found both the reflector IP addresses and the original destination IP addresses are scattered in the IP address space. The path backscatter AS tuples are plotted in Fig. 10. The reflector ASes and the original destination ASes are also scattered. This result shows the networks which are capable of generating path backscatter messages are very diversified. Besides, it implies a victim can suffer attacks from a large number of corners of the Internet.

3) Geolocations of Reflector IP Addresses: Fig. 11 plots the geolocations of the reflector IP addresses. The geolocation data of IP addresses here and later is got from IPInfoDB [37]. It can be found the reflectors are distributed all over the world. Their locations at least tell the locations are capable to generate ICMP error messages.

4) Statistics on Classes: Fig. 12 illustrates the number of path backscatter messages and IP tuples of each class. TIMXCEED_INTRANS and UNREACH_HOST cover a majority of all the messages and IP tuples. We use ‘Stub messages’ to denote the path backscatter messages whose source AS is a stub but the original destination is not the same AS. Such messages must be from the routers in the same AS as the spoofer. It should be noted that not only such messages are near the spoofers. The special classes listed in Section IV-C are also near the spoofers. We use ‘Stub IP tuple’ to denote such messages. It can be found such messages are non-trivial.

5) The Top Reflectors and Original Destinations: We analyzed the top 10,000 reflectors and original destinations which appear the most frequently in IP tuples. The CDF (Cumulative Distribution Function) of involved original destinations IP numbers of the top reflectors are plotted in Fig. 13(a), and the CDF of involved reflectors IP numbers of the top original destinations are plotted in Fig. 13(b). The result in Fig. 13(a) shows that a small number of reflectors forwarded spoofing traffic to a large number of original destinations. We do not claim the reflectors are the attackers, but they are very likely near the spoofers because the spoofing traffic traverse them to reach a large number
of destinations (though they can be the gateway of a large network). There may be a convenient assumption that the top ref-lectors may distribute in a few areas, but they are not. Fig. 14 plots the geolocations of the top 10000 reflectors. It can be found they are widely distributed.

The result in Fig. 13(b) shows a small number of addresses are facing attacks from a large number of corners. However, we found the top original destinations are mostly private addresses. Anyway, the result implies the number of potential attackers are large. Other than private original destination addresses, the top original destinations did not involve a large number of reflectors. These numbers are far less than the total number of reflectors. This implies the attackers in each attack are not many. This result coincides with the report from Arbor.

Fig. 15 shows the geolocation of the top non-private original destination and the involved reflectors. The total number of reflector IP addresses are 625 (assigned in 10 ASes). This number is just about 0.2% of the total number of reflectors. Besides, it can be found the reflectors are not distributed all over the world.

6) On the Filtering of Path Backscatter Messages: We found two NAT reflectors (both in China). However, we did not filter out any path backscatter message based on the mechanism proposed in section III-D. We found that, other than the cases in which the original destination is private address, the largest number of IP tuples whose original destinations are same is less than 650. Considering an attacker can easily generate much more tuples, it implies the dataset can be trustful to some extend. Anyway, because the filtering mechanism certainly can not found all the forged path backscatter messages, we do not claim all the messages used in our experiments are trustable. However, we think this fault will not make the whole solution worthless.

B. Evaluation on PIT

Though there are huge amounts of path backscatter messages generated, their generation does not have a certain probability. Thus, it is impossible to evaluate PIT similar as the other IP traceback mechanisms which have stable packet marking/ICMP generation probability. For this reason, we do not evaluate how well PIT will work in each attack. To exclude the uncertain factors of path backscatter message generation, we evaluate the possibility of locating the attacker after we get a random path backscatter (reflector, original destination) tuple. To achieve this, we make the following assumptions on IP spoofing attacks and path backscatter generation:

1) Assumption I: the locations of attackers are random;
2) Assumption II: attackers choose random destinations to send spoofing packets;
3) Assumption III: the captured (reflector, original destination) tuple is generated on a random hop from the attacker to the original destination.

We evaluate the mechanisms proposed in section IV-B based on these assumptions. We begin with deducing the probability of accurate locating. We compare the estimated result based on the deduction with the simulation result. The simulations are performed as follows: we choose a random node to be attacker, choose a random destination, and choose a random hop to be the reflector; then we check the (reflector, original destination) tuple can be used to accurately locate the attacker based on the proposed mechanisms. In section V-B1 and V-B2, we present the probability of accurate locating the attacker; and in section V-B3, we present the distribution of suspect set size. We found that, without considering the occasional factors of path backscatter message generation, the effectiveness of the mechanisms are actually determined by the structure of the networks. Though very limited information can be used in the tracing, the attacker tracking is found to be effective largely due to the power-law structure of the networks.

1) The Probability of Accurate Locating on Loop-Free Assumption: Based on the Loop-free assumption, to accurately locate the attacker from a path backscatter message \( (r, od) \), there are three conditions:

1) \( LF-C1 \): the degree of the attacker \( a \) is 1;
2) \( LF-C2 \): \( od \) is not \( a \);
3) \( LF-C3 \): \( r \) is \( a \).

Based on the Assumption I, the probability of \( LF - C1 \) is equal to the ratio of the network nodes whose degree is 1. To estimate this probability, we introduce the power law of degree distribution from [38],

\[
fd \propto d^O
\]

where \( fd \) is the frequency of degree \( d \), and \( O \) is the outdegree exponent. Transform it to

\[
fd = \lambda d^O + b_d
\]

where \( \lambda \) and \( b_d \) are two constants. Then, \( f_1 = \lambda + b_d \).
Based on the Assumption II, the probability of $LF - C2$ is simply $(N - 1)/N$. Based on the Assumption III, the probability of $LF - C3$ is equal to $1/(1 + len(path(a, od)))$. Because $a$ and $od$ are random chosen, the expectation of $len(path(a, od))$ is the effective diameter of the network, $\delta_{ef}$ [38].

Based on the three assumptions, these conditions are mutually independent. Thus, the expectation of the probability of accurate locating the attacker is

$$E(P_{LF\text{-accurate}}) = \frac{N - 1}{N} \times \frac{\lambda + bd}{1 + \delta_{ef}}$$

This form gives some insight on the probability of accurate locating. If the power-law becomes stronger, $\lambda$ will get larger and $\delta_{ef}$ will get smaller. Then the probability of accurate locating will be larger.

We plot this estimated value and the corresponding simulation result for AS-level topologies of each year in Fig. 16. The probability is stably around 0.05.

We plot this value for networks in the topology zoo dataset in Fig. 17. It can be found in more than 40% of the topologies, the probability of accurate locating the attacker based on a random tuple is larger than 0.1.

These results show, because of the power-law structure of the networks, even just using the loose loop-free assumption on routing, the probability of accurate locating the attacker based on a random tuple is non-trivial.

2) The Probability of Accurate Locating Based on Valley-Free Assumption: Similarly, based on the valley-free assumption, to accurately locate the attacker from a path backscatter message $(r, od)$, there are three conditions:

1) $VF-C1$: the size of $Cone(a)$ is 1;
2) $VF-C2$: $od$ is not $a$;
3) $VF-C3$: $r$ is $a$.

However, there is no convenient power law on the distribution of customer cone size. Fig. 18 plots the CCDF (Complementary Cumulative Distribution Function) of the customer cone size based on the result from CAIDA in log-log scale. It can be found that, when the customer cone...
size is smaller than 1000, the CCDF is almost straight in the log-log plot; thus, the CCDF well follows power-law.

Based on this discovery, we define the **Customer Cone Power Law**:

**Definition (Customer Cone Power Law):** The CCDF (Complementary Cumulative Density Function) of customer cone size of a size follows power-law when the size is much smaller than the number of ASs:

\[
R_k \propto k^C
\]

where \( R_k \) denotes the ratio of ASs whose size of customer cone is larger than \( k \), and \( C \) is the customer cone exponent. Translate it to

\[
R_k = \gamma k^C + b_c
\]

where \( \gamma \) and \( b_c \) are two constants.

If the spoofer can be located exactly, the size of customer cone must be 1. The ratio of such ASs is \( 1 - R_1 = 1 - \gamma - b_c \).

Then, similarly, the probability of accurate locating is:

\[
E(P_{VF-accurate}) = \frac{N - 1}{N} \times \frac{(1 - \gamma - b_c)}{1 + \delta_{ef}}.
\]

We plot this value for AS-level topologies of each year and the simulation result in Fig. 19. This value is stably around 18%. It should be noted that PIT does not require deploying additional infrastructures. This traceback capability is achieved with trivial additional effort. Compared with zero traceback capability without PIT, this is a significant result.

3) **The Distribution of Suspect Set Size:** If the (reflector, original destination) tuple cannot be used to accurately locating the spoofer, the size of the suspect set size is meaningful. We present the distribution of the suspect set size in Fig. 20. It can be found a tuple will either determine a small suspect set, or be almost useless. Such a distribution is also caused by the power-law structure of the Internet.

C. Performing PIT on the Dataset

In this section, we present a number of meaningful results after performing PIT on the path backscatter dataset.

1) Where Are the Spoofers?: In this section, we present the locations of the spoofers captured. This result is achieved through combining the tracking mechanisms proposed in section IV.

The procedure is as follows. For each path backscatter message, at first we check whether it belongs to the special classes listed in section IV-C. If yes, the reflector should be near the attacker. We simply use the source AS of the message as the location of the spoofer. If the message does not belong to the types, it is mapped into an AS tuple. Determine whether the AS tuple can accurately locate the source AS of the attacker based on the mechanisms proposed in section IV-B. Because we perform tracking at the AS level, we only use the valley-free assumption which results in better tracking capability than the loop-free assumption. Then if the AS tuple can accurately locate the source AS of the message, the source AS of the spoofer is just this AS. Then we also use the source AS as the location of the spoofer. We do not further investigate the location of the spoofer inside the AS because we do not know the inner structure and address allocation in the ASes. However, at least the messages of the special classes listed in section IV-C can help locate the network of the spoofer.

We got 2788 ASes in which there are spoofers. 914 of them are located by the mechanisms in section IV-B, and 2148 are located based on the special classes of path backscatter messages. There are 274 ASes located by both mechanisms. The full list of the ASes can be fetched from http://tinyurl.com/lp959y4.

The captured ASes are only a small portion of all the ASes. We believe this result underestimated the total number of ASes with spoofers reside in. Considering the limitation of the backscatter collection capability of the CAIDA network telescope, the uncertainness of path backscatter generation and the available datasets we can access, we are not able to provide a complete list of all the ASes in which there are spoofers. Here we just present this partial result to illustrate the effectiveness of this proposed tracking mechanism. It can be the basis for further potential works.

Besides, it should be noted that the ASes with spoofers in are not the ASes which indulge spoofing. Actually, there are a number of path backscatter messages are generated because of the filtering performed by the ASes.
2) An Aggregated Attack: We performed tracking based on all the path backscatter messages whose original destination is 194.97.X.Y, which is assigned in AS5430. There are totally 13511 such path backscatter messages, and 30 ASes are located (15 of them are located by the special classes of path backscatter messages, and 19 of them are located based on the valley-free assumption. 4 of them can be located by both mechanisms). We plot them in the AS-level topology as Fig. 21. We make use of shortest valley-free path to connect the ASes with spoofers located and the victim AS.

This proposed mechanism certainly does not work in all the attacks and can not capture all the spoofers, but it does tell something about the spoofing attacks. At least, the luckiest victims are able to locate some of the spoofers. This is valuable until an AS-level traceback system is established.

VI. CONCLUSION

We try to dissipate the mist on the the locations of spoofers based on investigating the path backscatter messages. In this article, we proposed Passive IP Traceback (PIT) which tracks spoofers based on path backscatter messages and public available information. We illustrate causes, collection, and statistical results on path backscatter. We specified how to apply PIT when the topology and routing are both known, or the routing is unknown, or neither of them are known. We presented two effective algorithms to apply PIT in large scale networks and proved their correctness. We demonstrated the effectiveness of PIT based on deduction and simulation. We showed the captured locations of spoofers through applying PIT on the path backscatter dataset. These results can help further reveal IP spoofing, which has been studied for long but never well understood.

APPENDIX A

PROOF OF THEOREM 1

Proof (1) Necessity:=> In G, there are at least two loop-free paths P1, P2 from r to c which contains no identical verticals other than r and c. Otherwise, the removal of one of the two verticals c’ will break r and c. Because any path from r to od must pass c, then the removal of c’ will also break r and od. Besides, c’ must be contained in the path(r, od). Then c cannot be the first vertex whose removal breaks r and od.

In SG(r), there exists one loop-free path from any vertex to r, which has identical verticals with at most one of P1 and P2. It is because whenever a path comes across P1, it can travel upstream P1 to reach r without necessity of crossing P2; and vice versa. Then a loop-free path from any vertex v to c can be constructed through composing the loop-free path path(v, r) with either P1 or P2. Join this path with the segment from c to od in path(r, od), and a loop-free path from v to od can be constructed.

(2) Sufficiency<=: In SG(r), if a vertex v is no connected with r, then in G, c must be contained in path(v, r) and path(r, od). Thus there is a loop in any path from v to od passing r.

APPENDIX B

PROOF OF THEOREM 2

Proof (1) Necessity:=> Assume when od is not in Cone(r), v is not in Cone(r). Based on the valley-free assumption, the path from v to od passing r must a valley-free path.

Because v is not in Cone(r), the path from v to r can be: 1) a downhill path; 2) an uphill path followed by a downhill path; 3) an uphill path followed by a peer-peer edge; 4) a peer-peer edge followed by a downhill path; 5) an uphill path followed by a peer-peer edge, which is followed a downhill path.

Because od is not in Cone(r), the path from r to od can be: 1) an uphill path; 2) an uphill path followed by a downhill path; 3) an uphill path followed by a peer-peer edge; 4) a peer-peer edge followed by a downhill path; 5) an uphill path followed by a peer-peer edge, which is followed a downhill path.

However, any combination of the two paths be a valley-free path. Thus, the assumption that v is not in Cone(r) is invalid. It means v must be in Cone(r).

(2) Sufficiency<=: When od \notin Cone(r) and there is a valley-free path from v to od passing r, the path from r to od can be: 1) an uphill path; 2) an uphill path followed by a downhill path; 3) an uphill path followed by a peer-peer edge; 4) a peer-peer edge followed by a downhill path; 5) an uphill path followed by a peer-peer edge, which is followed a downhill path.

The path from v to r must be a valley-free path, thus, it can be 1) an uphill path; 2) a downhill path; 3) an uphill path followed by a downhill path; 4) an uphill path followed by a peer-to-peer edge; 5) a peer-to-peer edge followed by a downhill path; or 6) an uphill path followed by a peer-to-peer edge, which is followed by a downhill path.

However, only when the path from v to r is an uphill path, the combined path from v to od can be a valley-free path. Then v must be in cone(r).

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers and the editors for their valuable comments and suggestions to improve the quality of this paper. They are also grateful to Dan Massey and Bingyang Liu for reviewing the paper.

REFERENCES


Guang Yao received the Ph.D. degree in computer science from Tsinghua University, Beijing, China, in 2012, where he is currently a Post-Doctoral Researcher with the Network Research Center. His research interests include IP traceback and software defined networking.

Jun Bi (M’00–SM’14) received the B.S., M.S., and Ph.D. degrees in computer science from Tsinghua University, Beijing, China. He was a Post-Doctoral Scholar with the Department of High Speed Network, Bell Labs Research, Murray Hill, NJ, USA, and a Research Scientist with the Bell Labs Research Communication Science Division and the Bell Labs Advanced Communication Technologies Center. He is currently a Full Professor and the Director of the Network Architecture and IPv6 Research Laboratory with the Network Research Center, and a Ph.D. Supervisor with the Department of Computer Science, Tsinghua University. His research interests include network architecture, Internet routing and addressing, IPv6, and future Internet. He is also a Senior Member of the Association for Computing Machinery.

Athanasios V. Vasiilakos (M’00–SM’10) is currently a Professor with the University of Western Macedonia, Kozani, Greece. He has authored or coauthored over 200 technical papers in major international journals and conferences. He has authored or coauthored five books and 20 book chapters in the areas of communications. He has served as the General Chair, the Technical Program Committee (TPC) Chair, and a TPC Member (i.e., INFOCOM, SECON, and MOBIHOC) for many international conferences. He served as an Editor or a Guest Editor for many technical journals, such as the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, PART B, the IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE, the IEEE TRANSACTIONS ON COMPUTERS, the ACM Transactions on Autonomous and Adaptive Systems, the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, the IEEE Communications Magazine, the ACM Wireless Networks (Springer), and the ACM Mobile Networks and Applications (Springer). He is a Founding Editor-in-Chief of the International Journal of Autonomous and Adaptive Communication Systems and the International Journal of Arts and Technology. He is also the General Chair of the Council of Computing of the European Alliances for Innovation.