Pioneering Technologies
for a Better Internet

Cs3, Inc.

5777 W. Century Blvd.
Suite 1185
Los Angeles, CA 90045-5600

Phone: 310-337-3013
Fax: 310-337-3012
Email: info@cs3-inc.com

Changing IP to Eliminate Source Forgery

Donald Cohen

K. Narayanaswamy

Fred Cohen

Abstract

Source address forgery is widely recognized as one of the biggest security problems in the Internet today. This White Paper describes an enhancement of Internet Protocol (IP), called Path Enhanced IP (PEIP), designed to eliminate source forgery. Specifically, PEIP makes it possible to recognize packets with forged source addresses, see where they come from, and, if necessary, even filter them. Ingress filtering, the much recommended but little used partial solution to source forgery, is still beneficial in the presence of PEIP. Further, PEIP actually improves prospects for enforcement of ingress filtering. Elimination of source forgery makes PEIP a more solid foundation than IP for a robust and secure Internet infrastructure.

Contents

1. Source Address Forgery and Its Dangers

Communication in the Internet works by sending packets from one place to another. Each packet, like a post card, contains a source address and a destination address. The Internet fulfills the role of the post office, delivering packets to their specified destination addresses. It is interesting to notice that *only* the destination address is used to deliver the packet. In most cases, however, the sender wants the destination to reply. The source address is used by the destination to address the reply.

Unfortunately, considerable mischief can be caused by sending packets with incorrect source addresses. First, it is very likely that those sending such unwanted messages would also like to avoid being identified by the recipients. Even worse, a recipient who believes the forged source address will blame the owner of that address for the unwanted message.

Some of the worst attacks in the Internet today involve sending packets that cause automatic replies. Typically, in this case, neither the party that receives the original packet nor the party that receives the reply would object to a few such packets, but the attacker arranges for them to get huge numbers. Each feels like he is being attacked by the other. Alternatively, a large number of places are sent a smaller number of packets and the replies all converge on a victim who sees an attack that appears to come from a large number of places. Even if the attack is coming from a large number of places, that number can be made to appear much larger by reflecting the packets off many innocent intermediaries.

2. Ingress Filtering

The commonly recommended solution is called *ingress filtering*. In the post-office analogy, ingress filtering corresponds to the local post office only accepting letters with its own zip code in the return address. A potential attacker can still successfully use valid return addresses that belong to his neighbors, or even non-existent addresses with that zip code. However, if these letters cause trouble, the range of possible suspects is relatively small. If the attacker sends many such letters, the local post office is likely to find out that the letters came from him.

All of the above statements hold for the Internet as well as the post office. The local post office corresponds to an ISP. The ISP knows that all of the packets it sends out should have source addresses in a given range. Ingress filtering simply refuses to forward those packets with source addresses outside that range. (Note that the post office actually does have something just as good as this. The postmark shows where a letter entered the system. The recipient can compare the return address to that postmark.) This suggestion has appeared earlier in the literature [Cohen 2]. It is also currently advanced in many places, such as RFC2267 and RFC2827 available from rfc-editor.org, CERT® Advisory CA-1996-21 and SANS - DDoS Roadmap: Steps 1 & 2 NOW!.

There are a few problems with ingress filtering. First, as noted, it is still hard to tell when an attacker forges the address of his neighbor. More important, the Internet, unlike the post office, has no central authority that can force all of the ISPs to check the source addresses of outgoing packets. Finally, this extra effort does not really help the ISP that exerts the effort. It helps the rest of the world by preventing source forgery originating from that ISP, whereas the customers who pay the ISP for service get no direct benefit. In light of this, perhaps it is not so surprising that, in spite of all of the recommendations above (and their appeals to the spirit of cooperation), few, if any, ISPs actually do ingress filtering.

3. Discerning True Packet Sources

Unlike the post office, the Internet consists of a large number of different carriers who forward packets to each other. If things work correctly, then each forward operation brings a packet closer to its destination until it finally arrives there. When a packet arrives, each forwarder can normally tell who sent it the packet, but not where that party, in turn, got the packet, or anything more about the path the packet took through the different carriers. Our proposal is for each forwarder to add to the packet an indication of who gave the packet to it. Each packet will then arrive with a complete forwarding path.

Again, this is not a new idea [Cohen 1]. The advantage of this scheme is that the path is not controlled by the sender. A forger is now in the position of writing a return address of New York when the receiver can plainly read a postmark that says Chicago! This is really a proposal to change the communication protocol used in the Internet. The new protocol is viewed as an enhancement of the Internet Protocol (IP), and is referred to as **Path Enhanced IP**, or **PEIP**.

The "indication" above of where the packet came from does not have to be an IP address. That would take much more space than necessary, space that could otherwise be used for real data. In PEIP, space is saved by encoding the paths. The encoding scheme and some additional reasons to save space are mentioned below.

A forwarder that can receive packets from ten different places will add to the path a number between one and ten. This, naturally, means that receiver needs a way to decode the path. However, a separate protocol will be needed to decode the path into a sequence of IP addresses. This protocol requires each machine along the path to take the data that it (supposedly) added, and find the IP address of the neighbor for which it would have added that data. The mappings between neighbors and added data are expected to remain reasonably stable. In other words, within reason, it should be possible to trace the path of a packet that was received in the past.

For reasons that will become clear below, it is desirable for the path to include one explicit IP address, generally the address of the first router to forward the packet. The rule is that if a forwarder cannot be sure who gave it the packet, as is typically the case when it comes out of a LAN, [EndNote 1] then it must discard any path that came with the packet. In effect, the forwarder is taking responsibility for the packet. The originator of a packet likewise should send it with an empty path, meaning he did not get it from anywhere else.

Note that the final recipient is typically in a LAN but he will not discard the path. He will use it to see where the packet come from. The recipient that puts the first element in a path (which implies that he is sure who sent him the packet) starts the path with the actual IP address of the sender. In effect there are two cases.

- A packet that comes from a LAN goes to the router connected to the LAN which forwards it with an empty path. The recipient then starts the path with the IP address of that router, which is the one that allowed the packet into the network.
- A packet that is created by a router is sent with an empty path. The recipient then puts the IP address of that router in the path indicating that the packet is the responsibility of the originating router.

We advocate that ingress filtering be not only universally adopted, but actually enforced. This involves both a technical and a social component. Source tracing is the technical component. Its role is to make apparent to all where forged packets enter the network and who was foolish enough to trust those places. The social component involves shunning those places that forward forged packets. Source tracing provides the accountability that both justifies denying service to the guilty party (since his guilt is evident) and forces his provider to take this action (otherwise, he appears to act as an accomplice).

Ideally, the allowable source addresses should be part of the agreements between ISPs, e.g., ISP A agrees to accept and forward from ISP B traffic with the following IP source addresses. Then both ISPs should filter traffic with non-conforming source addresses.

If necessary, the paths themselves can be used to filter offending traffic. Suppose Alice gets traffic from Bob who gets it from Charlie. Alice sees that this traffic has forged source addresses (or is objectionable to her for some other reason). She can complain to Bob and ask him to stop accepting packets from Charlie, or at least stop forwarding them to her. However, even before she can contact Bob, and later, even if Bob refuses, Alice can filter the packets from Bob with paths indicating that he got them from Charlie. In general this filtering can be done anywhere along the path. The explicit IP address at the end of the path will turn out to be especially useful for this kind of filtering.

4. Need For Two Paths

The paths above show where a packet came from. However, there is an important class of attacks for which additional data would be very useful. These attacks involve sending packets with forged source addresses to an intermediate host, causing it to reply to those forged source addresses. For instance, if attacker Albert sends a request to intermediate host Harry with the source address of victim Victor, Harry will have no reason to suspect that the source address is forged. If Harry simply replies (or, more important, if he and a million others reply) then Victor gets a reply from Harry (and a million others like him). Victor can tell where the replies came from but this does not help him. What would be more useful is the source of the original request packet.

For this reason, we propose that all replies to packets that could easily have been forged contain the path that the original packet took from its sender to its receiver, who is now replying. That is, Harry sends the path from Albert to Harry along with the reply. Victor now gets three things: the unsolicited reply, the path from Harry to Victor and the path from Albert to Harry. Victor can now not only trace back to Harry, but, by concatenating the path from Albert to Harry to the path from Harry to Victor, he can trace the path all the way back to Albert! (In this case, the explicit IP address at the end of the path from Albert to Harry is likely to be the most important piece of data.) Of course, it would be preferable for Albert's ISP, Isaac, to find and punish Albert. It is also reasonable to blame Isaac for failing to filter packets with Victor's address as their source. In fact, one would hope that Isaac's service provider, in turn, will threaten to stop carrying traffic for Isaac unless he starts filtering his packets. And so on...

Although many different types of packet satisfy the description of those that should send double paths, most of the packets actually sent in the Internet do not need to. In particular, most TCP packets reply to (acknowledge) a previous packet that could not have been sent by an attacker unless he either made a very lucky guess or happened to control a router that forwarded an earlier packet in the same connection (in which case he is already in position to do a lot of damage). These packets therefore need not send a second path. On the other hand, a TCP packet that results in a *no-such-connection* error response could be sent by an attacker using a forged source address, so the *no-such-connection* response should contain the extra path. Similarly a SYN packet could contain a forged source address, so the SYN-ACK response should contain an extra path.

5. Use of Explicit IP Addresses

The attack described in Section 4. above motivates the inclusion of the explicit IP address at the end of the path. Albert sends packets to a huge number of intermediate hosts, causing them all to reply to Victor. Victor sees unsolicited packets coming from all sorts of addresses by all sorts of paths. Without the explicit IP address he can see the source of the attack by tracing the paths of a few unsoliticed packets. What he cannot easily do is recognize (and thereby filter, or ask his upstream providers to filter) all of the paths from Albert via all possible intermediaries. On the other hand, it is easy to recognize the packets containing secondary paths ending with a particular IP address, in this case that of the router that allowed Albert's packets into the network.

The inclusion of one explicit IP address is a compromise based on the assumption that, routers are much less likely than normal hosts to be controlled by attackers. If Albert controlled a router he could have it send packets with arbitrary paths, including random IP addresses at the ends. Then Victor would no longer be able to easily identify the attack packets. However, if paths were completely unencoded, the IP address of the attacking router would be in the secondary path of every attack packet, and could therefore be used to filter those packets. The cost, of course, is the space required by unencoded paths, 100 bytes in IPv4 and 400 bytes in IPv6 for 25 hops. With encoded paths the only way to stop such an attack is to get the neighbors of the attacking router to stop forwarding its packets. The problem of identifying the attacking router is addressed in Section 8.1..

The packets arriving with secondary paths should be replies to packets sent by the recipient. Therefore the secondary paths ought to end with the addresses of routers that are neighbors of that recipient. This should make it easy, at least for places close enough to the ultimate destination, to filter out packets that reply to requests with forged source addresses. However, we hope that source forgery will be sought out and punished, not just filtered near the victim.

As an aside, it is interesting to notice that a single ping followed by a single trace operation would generate the equivalent of a round trip traceroute.

6. How Expensive is PEIP?

The longest paths in the Internet are currently about 25 hops. The average is actually much less. The routers that forward packets are typically connected to no more than 16 other routers. Therefore a typical hop should take no more than 4 bits. This gives a total of about 16 bytes for the longest paths in IPv4 (including the 4 byte explicit address) and 28 bytes in IPv6 (where the explicit address is 16 bytes).

Of course, in packets with an extra path, the expense could be twice as high. However, as noted above, these packets make up a small fraction of the traffic in the Internet. To give an idea of the value of the bandwidth being used, it is relevant to mention that the smallest possible IPv6 header is 40 bytes, whereas the smallest possible IPv4 header is 20 bytes. Most IPv4 headers are actually the minumum length. Anyone who wants to move from IPv4 to IPv6 therefore must be willing to pay 20 bytes per packet.

The time it takes a router to add its data to the path is a small constant. This should pose not a serious problem. If expanding a packet is problematic for specific routers, it would be possible to pre-allocate space. A more serious problem is that this extra data might require fragmentation. For non-attack traffic this does not seem like a major problem. TCP traffic, which comprises most of the traffic in the Internet, avoids this problem by using non-fragmentable packets to find a *Path MTU*. Attack traffic is discussed below.

A reasonable question is what maximum size of paths must be supported. Both IPv4 and IPv6 limit paths to 255 hops. As noted above, this is far more than any real paths. Of course, legitimate paths must not be cut off since that prevents source tracing. On the other hand, there are good reasons to limit the length to the maximum realistic path length. Something in the range of 30 hops or 16 bytes (for IPv4) seems like a reasonable limit.

7. Problems Caused by PEIP

This section represents an attempt to anticipate problems that might be caused by adoption of PEIP. Feedback from Beta testing of PEIP implementations and reviewers of PEIP documents will result in modifications to this section.

7.1. "Small Packet" Networks

The term "small packet" networks is used in RFC791 (IPv4) where it discusses fragmentation of packets. Clearly IP requires a lower level transport mechanism that is capable of carrying packets of some reasonable size. In particular, fragmentation would not help if the maximum packet size is less than the size of the IP header, since every fragment has to carry a header. It is preferable for every fragment to also carry a path. This effectively increases the minimal requirement by the maximum path size. That is one reason for keeping this maximum path size within reasonable bounds, i.e., far less than 255 bytes. This does not seem to be a problem with current or future technology.

RFC791 (IPv4) says "All hosts must be prepared to accept datagrams of up to 576 octets (whether they arrive whole or in fragments)." RFC2460 (IPv6) says "IPv6 requires that every link in the Internet have an MTU of 1280 octets or greater." It is not clear what might go wrong if these requirements were violated. Including the path in the data to be sent within these limits might result in failure of some other feature that requires that much data of its own. The alternative is to raise these limits to include the maximum amount of path data. That means that some implementations that satisfy the current requirements would fail to satisfy the new ones.

7.2. PMTU Discovery

The PMTU (Path Maximum Transmission Unit) in both IPv4 and IPv6 is defined as the minimum MTU of all links in a path. The discovery algorithm specified in RFC1191 (for IPv6) and RFC1981 (for IPv6) is indeed a straight forward algorithm for finding that quantity. Unfortunately, that is not what is needed if one is going to extend the path at each hop. The result will be that, e.g., a sender tries to send a packet of length 999 and gets back an error response complaining that the link can only send packets up to size 1004 ! However, in both cases the error response includes at least part of the original packet that provoked the response. If this includes enough data to compute both the length of the original IP packet and the length of the path at that link, then the sender will be able to adjust the PMTU correctly.

The above is one of several considerations that influence the choice of PEIP format. Since the error response, like other ICMP replies, includes the beginning of the packet that causes the reply, it would be convenient to put the path at the beginning of the packet and (re)define the data returned by ICMP to include this path. Another motivation for limiting the path length is that these ICMP replies carry only a limited amount of data from the original packet. The space devoted to the path is therefore deducted from that available for other data.

It is interesting to note in passing that paths would be useful to detect (and thereby defeat) the attacks mentioned in RFC1981 in which the attacker sends "Packet Too Big" messages.

8. Vulnerabilities of PEIP

The cases that have been recognized and analyzed are described below. Feedback from Beta users of PEIP implementations and reviewers of PEIP documents will result in modifications to this section.

8.1. Path Forgery

A key claimed advantage of PEIP is that the path is not controlled by the sender. Of course, this is not entirely true because paths can be forged too. While the sender cannot control the path that is recorded after the packet leaves him, he is able to control the path by which the packet supposedly reached him in the first place. That is, assuming no other router on the way alters the path, the recipient can definitely tell that the packet came via the sender, but the sender can, for whatever reason, claim that he got it from somewhere else by altering that part of the path.

There are really two cases to consider. The "normal" case is that an attacker controls a machine at the "edge" of the network. This turns out to be the easy case, as shown below. The more dangerous case is that the attacker controls a router that forwards traffic from many different places. He could then alter the paths of any packets forwarded by that router, or for that matter, manufacture new packets at that router with forged paths.

In many cases (probably the vast majority) it is easy to tell that this information is falsified. Suppose Alice traces the path of a packet she got from Bob. Bob says he got it from Charlie. Now suppose Alice sends a message to Charlie and he replies. One would expect that his reply would be marked with the same path as the original packet. If it is not then either someone forwarded differently than before (a different route), someone lied in answering the trace request, or someone forged the path of the original packet. In this case Alice knows that she got the original from Bob, so Bob is the only suspect. If Bob does not want to be caught by that sort of check then he has to claim that the packet at least came from someone who, given a packet addressed to Alice, would have forwarded it to Bob.

Suppose Bob can find such a neighbor. In fact, suppose it is actually Charlie. Bob can then send Alice packets that appear to come from Charlie. If Alice does not like these packets she cannot be sure whether they really come from Bob or Charlie. She can complain to both of them. She can also tell them both that she will filter out all packets she gets from Bob with a path indicating that they came to Bob from Charlie. Bob has now denied the service of communication from Charlie to Alice. Of course, he could have done that anyway by simply not forwarding packets from Charlie to Alice. The solution is for Charlie to stop sending his packets for Alice through Bob. He has to find a new path to Alice. If Bob now sends Alice a packet claiming to come from Charlie, the previously described check shows Alice that Bob is the forger, since replies from Charlie no longer come through Bob.

Of course, it is hoped that the case where a router is controlled by an attacker is very rare. Attackers who control routers can very likely cause even worse damage than forging source addresses. More likely, the attacker controls his own machine inside a LAN. He can try to send a packet with an arbitrary path, but the router that forwards that packet out of the LAN is required to discard that path. (Of course, it should also filter the packet if the source address is not inside the LAN.) Even if the router fails to do this, the check above will always show that this path is a forgery. The reason is that nobody should be forwarding packets into the LAN in order to reach a location outside the LAN. No matter where (outside the LAN) the attacker claims he got the packet, the replies from that place will come back along a different route.

It is, of course, possible to forward outside traffic through a LAN. The fact that this makes it impossible to reliably trace the source of such packets seems sufficient justification for disallowing this. Technically, the router that accepts such packets must take the responsibility for those packets. It is supposed to do this by deleting the paths coming from the LAN. The alternative is to trust the machines in the LAN to provide accurate paths. If they prove untrustworthy then the router that accepts their paths will be seen to be cheating, which is more than adequate cause for shunning that router.

8.2. Secondary Path Forgery

Notice that the packets with two paths mostly originate from hosts, and these hosts might well be controlled by attackers. These attackers might therefore forge the second path. Even easier, they could forge both a source address and a (primary) path in a packet sent to a host in their own LAN. That host would then reply to the forged source address with the attacker's forged path as the second path. This is regarded as secondary, rather than primary, path forgery since the ultimate objective is to send a packet with an incorrect secondary path. This sort of attack is relatively easy to counter by simply filtering the offending packets, all of which come from the network of the attacker.

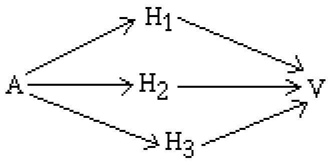
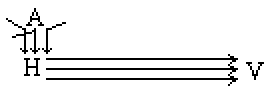
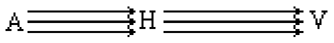
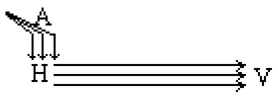
For the sake of completeness, it is also possible to attack other machines in one's own LAN with forged primary or secondardy paths. Of course, the administrator who is able to look at the traffic coming into or going out of the LAN will see immediately that these attacks are coming from the inside.

8.3. Example Attacks

An attacker who controls routers can simulate any of the attacks below. The means of identifying the attacking router was described above. That class of attack will be disregarded below. Instead, other attacks will be examined, and focus placed on how to distinguish one from another and how to filter the attack packets. It is not always possible to tell just from the paths in the attack packets who the attacker is. That is, one kind of attack can simulate another. The attack techniques considered are:

- **Source**: forged source address (allowed by a router with ingress filtering misconfigured; failure to delete paths from the packets where this is required is regarded as worse than misconfigured, but actually attacking) Source forgery without use of an intermediary to reply to the forged source address is ignored since this seems comparatively trivial.
- **Path2**: forged secondary paths (including forged paths inside a LAN) In this case there is no intermediary (or it is in the attacker's LAN).
- **Slaves**: control over a large number of slaves to do the above.

Attackers, intermediate hosts and victims will always be labeled A, H and V, with subscripts to indicate multiple instances.

Attack	**Diagram**	**Paths at victim**	**Filter**
1. Source		A ... H_1; H_1 ... V A ... H_2; H_2 ... V ...	A as source of secondary path
A sends packets to many different hosts, all with V's address as the source. V sees a lot of different double paths, but all originate from A.			
2. Path2		x ... ; H ... V y ... ; H ... V ...	H as source of primary path
A sends packets with many different (primary) paths and V's address as the source to host H on his own LAN. (Equivalently, he could manufacture replies with forged second paths.) V sees many different secondary paths, but all packets share the same primary path from A's LAN.			
3. Restricted Source		A ... H; H ... V A ... H; H ... V ...	A as source of secondary path OR H as source of primary path
Like Attack 1 but the same intermediate host is always used.			
4. Restricted Path2		x ... H; H ... V x ... H; H ... V ...	x as source of secondary path OR H as source of primary path
Like Attack 2 but the same fake path (from possibly fake address x) is always used. The interesting point is that A3 and A4 are not distinguishable without contacting places that are (supposedly) sending the attack packets to H.			

Slaves can give rise to much more ambiguity. (The diagrams, unfortunately, start to get unwieldy.) In general the ambiguity arises from restricting the attack so that V gets the same sort of paths as would arise from a different attack, or of course, a restriction of another attack. These restrictions do not make it any harder to filter the attacks. Rather they show that it will require additional communication in order to be sure about the source of the attack, and therefore in order to punish it.

Slaves doing variants of attack 1

As a starting point, the slaves could all do attack 1. This attack is straight forward to filter, but more inconvenient since it has to be done for each slave. If efforts to enforce ingress filtering meet with success, this sort of attack will become very difficult to execute and will be effectively limited to a small number of slaves. As an example of a restriction, the slaves could all use the same set of intermediate hosts. This would appear to V the same as attack 2. However, after each slave has sent one such packet either the attack must end or the secondary paths must start repeating, which is different from attack 2. Another problem with this attack is that it is likely to catch the attention of H who now seems to qualify as the victim since he is suffering at least as much as V.

Unfortunately, the fact that the restricted attack 5 can be distinguished from attack 2 is not very useful, since a restriction of attack 2 looks the same as the restricted attack 5. In this case the attacker simply reuses the same set of forged secondary paths over and over. Yet another approach is for the slaves to do this restriction of attack 2.

Slaves doing variants of attack 2

The slaves could also do attack 2. In this case ingress filtering is no obstacle. This attack is again straight forward to filter but inconvenient since it has to be done for each slave. The obvious restrictions include the slaves doing attack 4, or all of the slaves using the same set of forged secondary paths. All of these result in sets of paths arriving at V that have multiple interpretations.

8.4. Fragmentation

The fact that packets grow along the forwarding path suggests that attackers could send packets that will require fragmentation somewhere along the path. This is not a problem in IPv6 since routers do not do such fragmentation. In IPv4, however, they do. In a region where all links have the same MTU this could be used to force one router to do a lot of fragmentation where it never had to do any before. This would cause it to use more resources of various sorts, which could cause it to drop more packets, or for that matter could cause it to crash. The commonsense solution is that routers should limit the rate at which they are willing to do this (and all other expensive operations) so that they drop the expensive packets if necessary rather than the others. This is not regarded as an additional requirement for a router, since routers should already do this.

8.5. Flooding the Source Tracing Facility

The fact that a new service is proposed for decoding paths opens up new avenues for attack. Routers have to be able to respond to these requests. An attacker might attempt to overload a router with such requests. Of course, this is nothing new. The attacker could send all sorts of other packets to routers. The obvious answer is that the routers have to limit the amount of resources they expend on these requests to ensure that they can still do their primary job. The fact one attacker might try to use up this limit sa as to deny service to others is also a familiar problem. The Cs3 solution to this and many other similar attacks is to use paths to allocate effort. This is described more fully in A Fair Service Approach to Defenses against Packet Flooding Attacks

9. Other PEIP Publications

The following additional documents are (or soon will be) available regarding PEIP.

- *Path Enhanced Internet Protocol (PEIP) Technical Specification*; (Cs3 Internal Technical Document). Please contact author for availability.
- *Path Enhanced Internet Protocol (PEIP) Adoption Issues*; White Paper in preparation. Please contact author for draft availability.
- A Fair-Service Approach to Defenses Against Packet Flooding Attacks describes how PEIP can be used as the basis to build defenses against Denial of Service attacks.

10. End Notes

End Note 1: Back to Reference

The term "LAN" is used in this context not to indicate anything about the physical diameter of the network but rather to indicate a shared medium in which it is possible for one sender to impersonate another.

11. References

[Cohen 1] Back to Reference

F. Cohen, Providing for Responsibility in a Global Information Infrastructure, IFIP-TC11, Computers and Security (under Feature Articles, Infosec Baseline Studies).

[Cohen 2] Back to Reference
F. Cohen, "Internet Holes - Eliminating IP Address Forgery" in "Network Security Magazine" (under Feature Articles, Managing Network Security).