May 2007

Geoff Huston

IPv6 Ping Pong

Some years back, in 2002, there was the SNMP security case, where it was exposed that most of the then deployed SNMP-managed devices (and there are a lot of such devices out there!) were vulnerable to attack, due to a poor implementation of the protocol handler in the SNMP engine. More recently, there was the attack on WiFi networks through vulnerabilities with use of the WEP encryption technology. And we saw the BGP reset attack due to a vulnerability in TCP window handling with spoofed TCP reset messages. This is of course a very small sample space drawn from a daunting set of exposed security vulnerabilities over the years, and the more general conclusion is that it's a foolhardy person who would proclaim any technology to be completely invulnerable to attack. A more prudent person would take the position that its vulnerabilities, whatever they may be, have yet to be discovered.

So now it's the turn of IPv6 to have a rather impressive security vulnerability exposed. At the CanSecWest conference in April 2007 there was a presentation titled "Fun with IPv6 routing headers" by Philippe Biondi and Arnaud Ebalard, and in this column I'd like to explore the nature of the v ulnerability that was been exposed in that presentation.

Source Routing Headers

We've often heard that IPv6 made a number of changes to the basic Internet Protocol to improve the position of network security. One of the basic changes has been the use of a large enough address field that Network Address Translators (NATs) should be obsolete in IPv6. This, in turn, allows end-to-end security technologies, such as IPSEC, to protect not only the payload of an IP packet, but also to operate in Authenticated Header mode, and allowing the source and destination addresses of the packet to be protected from alternation during the packet's transit through the network. However, that's not quite the complete story. IPv4 had the potential to add extension fields to the IP packet to support "strict source routing" and "loose source routing". In IPv4 the sender of the packet (the "source") could add the strict source routing header extension to the IP packet and then list in that header extension up to a further 9 IP addresses. The intent was to allow the packet's source to specify precisely the packet's path through the network. Using this header the packet had to be passed directly from the source to the device with the first IP address in the strict source routing heading extension, and this device had to pass the packet directly to the device with the second listed IP address, and so on. The last device in the strict source routing list had to pass the packet to the destination. In other words the strict source routing header extensions listed the precise path the packet had to traverse through the network. A variant of this was IPv4's loose source routing option, where rather than specifying the complete path from source to destination, the source could specify a sequence of 'landmark' addresses where the packet had to traverse, but was also permitted to be passed to other devices while being sent from one landmark address to the next. In other words the first phase was to route the packet to the first "landmark" source routing address, and then route the packet to the second address, and so on.

IPv6 has replicated this form of loose source routing in the definition of the Type 0 Routing Header.

Why was this header extension defined in IPv4?

Because it can be useful!

I must admit to have used this extension in a number of network diagnostic tools many years ago, as did many others at the time. What if you wanted to know if there was a current routing path between two remote locations A and B? Even if you can ping A and ping B this still does not tell you whether A can reach B. But if you generate a loose source routing packet with a header extension of A then B, and a source and destination of yourself, then if you receive the packet back you can confirm that there is a viable path from A to B. Equally you could use a loose source routing header and specify B then A to check connectivity in the reverse direction. If you extended this slightly and used a loose source routing header extension of A, then B, and a destination of A, then you could confirm that there is a viable path from A to B and from B to A with a single test (Figure 1).

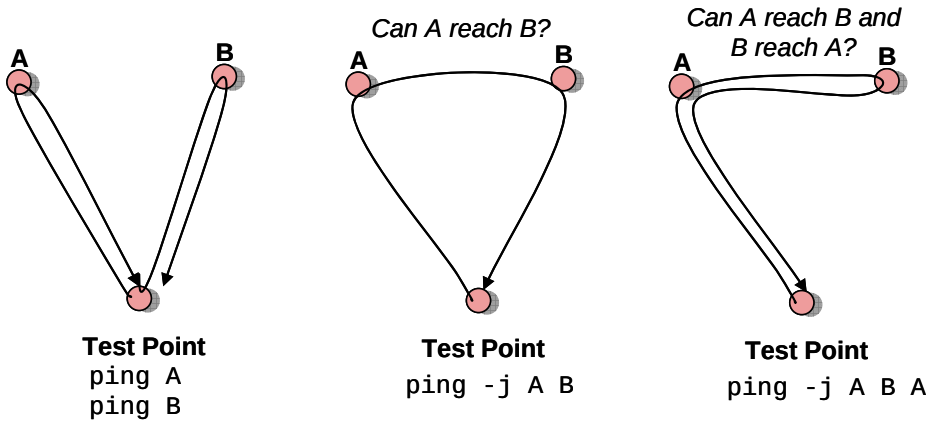


Figure 1 Loose Source Routing and Reachability Diagnostic Tests

So loose (and strict) source routing can be a useful diagnostic tool to test connectivity in a way that has far more capability than a simple ping or traceroute test. Loose source routing, particularly when applied to traceroute, can expose how others see routing paths, assisting in diagnosing various forms of routing anomalies.

What's happened to IPv4 Source Routing?

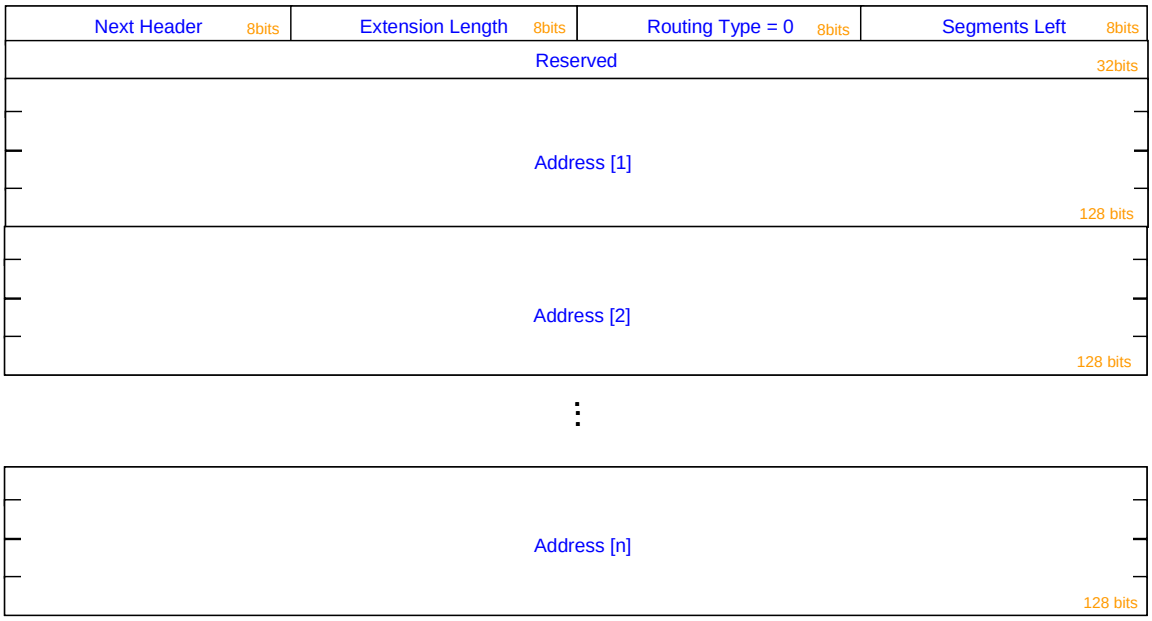
But all that was many years ago. These tools no longer work on today's production network, because most (and preferably all) network operators explicitly deny loose source routed packets, and that's on the whole a very good thing. There are a number of reasons for this, but most notably is the issue of exposing security vulnerabilities. If you have a firewall configuration that permits certain packets to enter your network, based on a set of rules relating to source and destination addresses in the packet's header, then loose source routing can readily defeat a simple-minded firewall. For example, if a network firewall permits a trusted remote network, 10.0.1.0/24 to access to the local network, then an attacker can attempt to gain access by generating a loose source routed packet, with a spoofed source address of 10.0.1.1 and adding a loose source routing header that specifies the attacker's host address. If the firewall permits the packet based on a trusted source address in the packet, then the attacker has in effect gained entry.

There are many ways to exploit loose source routing, and just too few reasons why the feature is useful, so the general approach these days is to configure IPv4 routers to deny packets with loose source routing headers and similarly configure firewalls to reject such packets.

IPv6 Type 0 Routing Headers

That was IPv4. What about IPv6?

IPv6 has a loose source routing capability defined as the type 0 routing header, as shown in Figure 2.




```
Next Header          8-bit selector.  Identifies the type of header
                     immediately following the Routing header.  Uses
                     the same values as the IPv4 Protocol field
                     [RFC-1700 et seq.].

Extension Length     8-bit unsigned integer.  Length of the Routing
                     header in 8-octet units, not including the first
                     8 octets.  For the Type 0 Routing header, Hdr
                     Ext Len is equal to two times the number of
                     addresses in the header.

Routing Type         0.

Segments Left        8-bit unsigned integer.  Number of route
                     segments remaining, i.e., number of explicitly
                     listed intermediate nodes still to be visited
                     before reaching the final destination.

Reserved             32-bit reserved field.  Initialized to zero for
                     transmission; ignored on reception.

Address[1..n]        Vector of 128-bit addresses, numbered 1 to n.
```

Figure 2 – IPv6 Routing Header

The IPv6 type 0 routing header indicates something subtly different to the IPv4 loose source routing header option. In the case of IPv6 the packet is delivered to the destination address as specified in the Ipv6 packet header. However the IPv6 destination node should now inspect the routing header of the packet, and if the type 0 routing header is present and if the Segments Left counter in the routing header is non-zero then the destination node is responsible for swapping the destination address with the next address in the routing header (as pointed to by the Segments Left counter), decrementing the Segments Left counter, and forwarding the packet onward to this next destination which is now in the IPv6 destination address of the IPv6 packet header. This mode of operation is indicated in Figure 3.

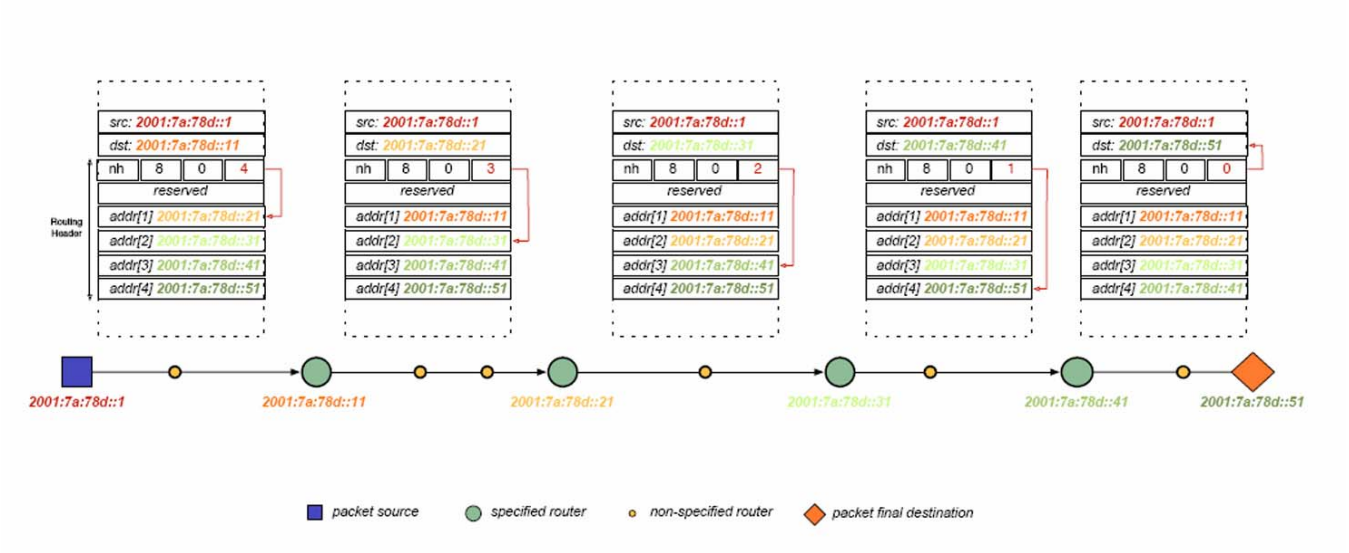


Figure 3 – Loose Source Routing in IPv6 – [Biondi & Ebalard]

In the IPv4 loose source routing model these intermediate routing 'landmark' addresses are addresses of routers. However this is not necessarily the case with the IPv6 type 0 routing header. The IPv6 specification, RFC 2460, drops this distinction, and states that:

> The Routing header is used by an IPv6 source to list one or more
> intermediate nodes to be "visited" on the way to a packet's
> destination.

Now the use of the word "node" is interesting in this context. RFC2460 describes a "node" as a device that implements IPv6, inferring that both routers **and** hosts should process this routing header.

The next question is how many intermediate routing destinations can be specified in an IPv6 packet header. The IPv6 specification has text on this as well, specifying that:

> Each extension header should occur at most once, except for the
> Destination Options header which should occur at most twice (once
> before a Routing header and once before the upper-layer header).

So a "well behaved" IPv6 implementation should generate packets with no more than 1 single type 0 Routing Header, and where the number of intermediate routing destinations in an IPv6 packet is limited to 64 intermediate destinations. But here is where the principle of "being conservative in what you send but be liberal in what you accept" comes into play, and the IPv6 specification also states that:

> IPv6 nodes must accept and attempt to process extension headers in
> any order and occurring any number of times in the same packet

So if an IPv6 speaker decides not to be "well behaved" then it can generate packets with more than one type 0 routing header. As Biondi and Ebalard point out, this is "the bullet in the foot", because now a sender can include as many IPv6 Type 0 routing headers as can fit in a maximally-sized IPv6 packet. In a 1280 octet IPv6 packet there can be up to 77 such loose source routed headers, and as the packet size increases, so can the number of loose source routed addresses.

Source Routed Ping Pong

So the vulnerability here is one of amplification, where a single packet alone can generate a large number of packets within certain targeting paths within the network. The technique is relatively simple: the routing header contains a repetition of a pair of addresses of the form A B A B A B … If this A B pair were repeated 3 times then a single packet directed at A would traverse the path A B 3 times, and B A twice. If such packets

were generated at a total rate of 1 Mbps then the path between A and B would experience a total of 5Mbps of traffic (Figure 4). But if you can place, say, 100 address pairs in the packet, then the same 1Mbps of routing header type 0 packets will place a total load of 199Mbps of traffic on the path between A and B. A small number of attackers can attempt to generate a massive traffic load with this technique. As Arnaud Ebalard points out having 100 pairs of addresses in an IPv6 packet is unusual, except for uncommon MTU values (a packet size of approximately 3300 octets in this case), and, more conventionally, one could see 44 pairs of addresses for a more common 1500 octet IPv6 packet.

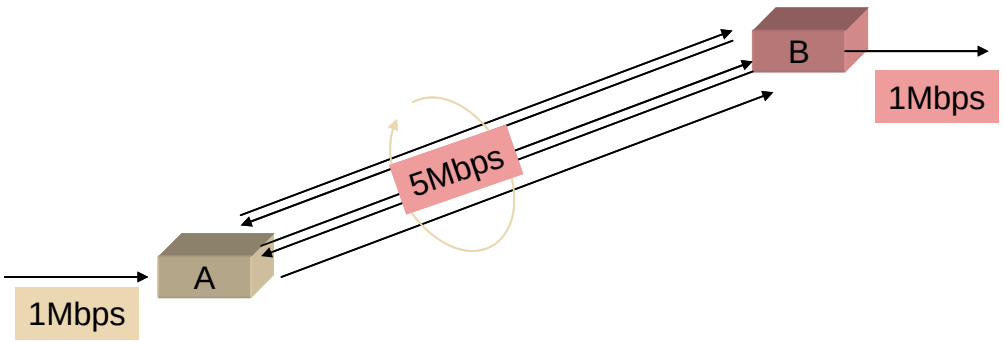


Figure 4- Path Loading via Loose Source Routing

These scenarios assume unidirectional traffic with a single packet stream. What if this is part of an orchestrated DDOS exchange? Each component of the DDOS attack may be located anywhere on the network and still be able to direct a traffic load to the network path between the same host pair. (Figure 5)

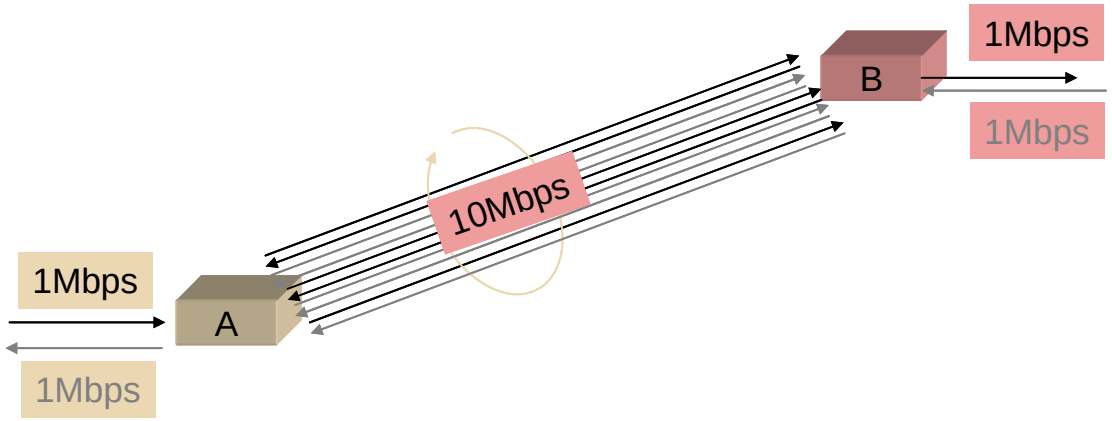


Figure 5- Path Loading via Loose Source Routing with multiple sources

While this loads up a path within the network, loose source routing does not allow for a concentrated amplification attack on a single host. Or can it? There is the potential for what has been termed “capacitance attacks”. In this approach a number of packets are generated with, say, 100 such pairs of addresses in the type 0 routing header, followed a sequence of packets with 99 pairs, then 98, and so on. With very careful timing of the packet sequences, all the packets in this A B loop can be released to the destination simultaneously, unleashing a short but intense traffic burst towards the target of the attack.

Now while this might appear to be an IPv6 network problem, what happens if A and B are connected via an IPv6-over-IPv4 tunnel?

So What's the Cure?

Well, as we already found out with IPv4 in production networks, source routing directives are, on balance, an undesirable protocol feature.

So the most reasonable cure is for IPv6 hosts and routers not to process a type 0 routing header. For some IPv6 host implementations this already happens, while for others its configurable, while other implementations will require the application of a source code patch.

The mantra of the implementor for Internet-based systems is the oft-quoted phrase from Jon Postel, who, in the Internet Protocol specification of 1981, cautions implementors to "*be liberal in what you accept.*" But perhaps we should not confuse liberality with imprudence, and one should also be mindful of what should very appropriately be rejected!

Further Reading:

The presentation of this vulnerability at CanSecWest 2007 is "*Fun with IPv6 routing headers*" by Philippe Biondi and Arnaud Ebalard: http://www.secdev.org/conf/IPv6_RH_security-csw07.pdf

Disclaimer

The above views do not necessarily represent the views or positions of the Asia Pacific Network Information Centre.

About the Author

GEOFF HUSTON holds a B.Sc. and a M.Sc. from the Australian National University. He has been closely involved with the development of the Internet for many years, particularly within Australia, where he was

responsible for the initial build of the Internet within the Australian academic and research sector. He is author of a number of Internet-related books, and is currently the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region. He was a member of the Internet Architecture Board from 1999 until 2005, and served on the Board of the Internet Society from 1992 until 2001.

http://www.potaroo.net