Internet Engineering Task Force
Internet Draft
Expiration Date: April 2004

P. Savola
CSC/FUNET

October 2003

Firewalling Considerations for IPv6

draft-savola-v6ops-firewalling-02.txt

Status of this Memo

Abstract

There are quite a few potential problems regarding firewalling or packet filtering in IPv6 environment. These include slight ambiguity in the IPv6 specification, problems parsing packets beyond unknown Extension Headers and Destination Options, and introduction of end-to-end encrypted traffic and peer-to-peer applications. There may also be need to extend packet matching to include some Extension Header or Destination Option fields. A number of often-raised, but not necessary relevant, issues are also summarized. This memo discusses these issues to raise awareness and proposes some tentative solutions or workarounds.

Table of Contents

1. Introduction

There are quite a few potential problems regarding firewalling or packet filtering in IPv6 environment. These include slight ambiguity in the IPv6 specification, problems parsing packets beyond unknown Extension Headers and Destination Options, and introduction of end-to-end encrypted traffic and peer-to-peer applications. There may also be need to extend packet matching to include some Extension Header or Destination Option fields. A number of often-raised, but not necessary relevant, issues are also summarized.

This memo discusses these issues to raise awareness and proposes some tentative solutions or workarounds. This document is not meant as a guide on how to set up IPv6 firewall policies (for example), but rather as a list of problematic issues to be considered by firewall developers, subject matter experts and those partipating in the IPv6 standardization effort.

On-link attacks using Neighbor Discovery are similar to ones available through IPv4 ARP, and not typically applicable to firewalls, and are therefore out of scope. A good summary of the issues is available [SENDREQ]. However, one should note that IPv4 ARP traffic is done using link-layer protocols, and is not generally seen (or blocked, even with a "deny all" policy) by firewalls; with IPv6, all such traffic is part of ICMPv6 Neighbor Discovery protocol suite, and thus more visible at the IP layer.

Implementation or policy-specific issues are mainly out of scope but partially touched on in section 7 about "non-problems"; these include e.g. issues of node-specific state creation (could be problematic if networks were brute-force scanned) and applicability of existing policies (e.g. blocking ICMPv6 would have very bad effects, particularly if certain link-local messages receive no special consideration).

In section 2, slightly ambiguous text in the IPv6 specification is discussed. In section 3, a syntactical problem with parsing unknown Extension Headers is pointed out. In section 4, a similar problem with Destination Options is discussed in the context of security policy. In section 5, implications of end-to-end encrypted traffic are considerated. In section 6, similar implications of peer-to-peer applications are mentioned. In section 7, a number of often-raised, but not necessary relevant, issues are summarized. In appendix A, some possibly useful packet matching extensions for IPv6 are brought up.

A possible generic denial-of-service attack using multicast and including amplification has also been noticed; as it is not firewall-specific, it is described (for the lack of a better place) with in Appendix B.

1.1. Terminology

In this document, the term "firewall" is used to mean any kind of packet filter; no special features (like statefullness or application-specific packet inspection) is assumed.

When considering firewalls, one should note that there are several ways to place and implement a firewall; in principle:

1. network firewall

2. network, user-controlled firewall

3. end-node, admin-controlled firewall

4. end-node firewall

Note that the second seems very rare, and the third is not really common yet. The reason why the entity which controls the firewall is explicitly mentioned is because it has significant implications on trust relations and which kind of solutions to the problems are possible.

The first is configured solely by an administrator, and placed in the network to block or pass the traffic of multiple nodes or network in an aggregated fashion.

The second is also configured by an administrator and placed in the network, but the user may be able to affect some policy decisions made in the firewall e.g. by some signalling protocol; ie. the policy of the firewall can, to some extent, be influenced by the end-nodes.

The third, also sometimes called a distributed firewall, is a firewall placed in the end-nodes, but controlled in some co-ordinated fashion by an administrator [DISFW].

The last is the typical end-node firewall, policy set by the end-user, or sometimes even the applications run on the end-node.

2. Ambiguous Text in the IPv6 Specification

2.1. The Problem

The [IPV6] specification forbids skipping over any of the headers before processing them or processing them at all before reaching the destination (section 4):

"With one exception, Extension Headers are not examined or processed by any node along a packet's delivery path, until the packet reaches the node (or each of the set of nodes, in the case of multicast) identified in the Destination Address field of the IPv6 header. There, normal demultiplexing on the Next Header field of the IPv6 header invokes the module to process the first Extension Header, or the upper-layer header if no Extension Header is present. The contents and semantics of each Extension Header determine whether or not to proceed to the next header. Therefore, Extension Headers must be processed strictly in the order they appear in the packet; a

receiver must not, for example, scan through a packet looking for a particular kind of Extension Header and process that header prior to processing all preceding ones."

And:

"If, as a result of processing a header, a node is required to proceed to the next header but the Next Header value in the current header is unrecognized by the node, it should discard the packet and send an ICMP Parameter Problem message to the source of the packet, with an ICMP Code value of 1 ("unrecognized Next Header type encountered") and the ICMP Pointer field containing the offset of the unrecognized value within the original packet. The same action should be taken if a node encounters a Next Header value of zero in any header other than an IPv6 header."

Similar applies to the specified Destination Options processing behaviour: if the Option Type has been specified so that the packet should not be processed further in the case of unrecognized options (ie. the highest-order two bits are not "00"), should the firewall also discard the packet and/or send ICMP Parameter Problem message back to the source?

Are these also to be done by intermediate nodes (which, by some definition, should not be processing Extension Headers or Destination Options Header with Hop-by-Hop options as an exception); this seems unlikely.

This wording clearly does not take into the account that there might be middleboxes, or non-final destinations, that could be processing the packet.

2.2. Possible Solutions

The correct behaviour must be made clear; the wording should be clarified. Clarifications might be needed at least on:

1. whether intermediate nodes should be taken into account in the text describing the header processing

2. intermediate nodes' behaviour when detecting unrecognized headers

It seems to be obvious that the firewalls will always inspect the headers, and in whichever order they want; see the next sections for descriptions of the specific problems.

3. Parsing Extension Header Chains

3.1. The Problem

IPv4 [RFC1122] [RFC1812] silently ignores options it does not recognize; options have a specific, pre-defined format. IPv6 Extension Headers are structured differently: the header format can change, and generally it is not possible to parse the header, or proceed to the following Extension Headers unless the processing of the previous header has been implemented.

The above is problematic as it is often the case that a packet filter will want to examine the terminal headers, e.g. TCP or UDP. That is not possible if there is a problem processing any one of the preceding headers.

Skipping over unknown headers and letting the packet through might be dangerous, if the unknown header would significantly change how the packet would be interpreted by the end-node.

One should note that all the currently defined Extension Headers, except Fragmentation, are encoded in the Type, Length, Value (TLV) -notation.

3.2. Possible Solutions

In the generic case, even ignoring the IPv6 specification, unknown headers cannot be skipped over except by making some very wild guesses of the content. Thus the solutions (or work-arounds) are:

1. always keep the packet filter up-to-date, so that it can parse all types of Extension Headers,

2. never introduce new Extension Headers, except possibly in a very controlled manner; use Destination Options instead, or

3. standardize the format (for at least the first N bytes including at least the length and the next header value) of possibly later specified new Extension Headers (for example, that all the new ones must be in TLV format), so that skipping over headers could be possible.

The first is not a workable solution in a generic case at least if it's expected that new Extension Headers should be introducable: the lifetime of firewall devices and software seems to be much longer than one would expect. For example, it is good to consider the case of buggy firewalls and ECN support [ECN]: even though software fixes may have been available for a long time, upgrades have not taken

place, hindering the deployment of new technology. It seems that keeping up with Extension Headers is not possible: software firewalls and their patch cycle are a problem big enough already, without considering hardware firewalls which could need new hardware implementations as well.

The second seems quite a bleak work-around, but as currently specified, there is little choice; most (if not all) new features can probably be implemented using Destination Options. However, it's still good to document and understand this deployment and specification deadlock.

The third might be doable but it would require some standardization effort.

4. Parsing Unknown Destination Options and Security Policy

4.1. The Problem

Similar to the above, Destination Options may also include unknown options. However, the options are encoded in the TLV-format. So, skipping over unknown options is technically possible.

However, especially in a very controlled environments, where a firewall may implement a strict security policy, it may be desirable to reject any packets whose options the firewall does not recognize (which may cause the end-nodes to do something that has not been anticipated in the security policy controlled by the firewall).

Skipping over unknown destination options and letting the packet through might be dangerous, if the unknown option would significantly change how the packet would be interpreted by the end-node.

4.2. Possible Solutions

No protocol action seems to be necessary provided that the implementation would not, in this case, send ICMPv6 messages or discard packets upon receiving an unknown header.

However, it may be desirable for firewall implementations to have a feature controlling the handling behaviour of unrecognized Destination Options.

5. Firewalls and End-to-End IPsec-encrypted ESP-traffic

5.1. The Problem

With the promise of the restoration of end-to-end transparency, and if at least some of the challenges for implementing Public Key Infrastructures are worked around, it may be possible that the amount of end-to-end encrypted traffic will increase enermously. The traffic is likely to be encrypted using IPsec.

In this case, on-the-path observers (such as a firewall) do not have the possibility to examine the usually critical headers (such as TCP/UDP). This may result in an administrative decision to disable IPsec-encrypted traffic by filtering it out completely, as the end-nodes' adherence to the security policies cannot be verified.

5.2. Possible Solutions

It would be desirable, if the users wish to do so, be able to have the firewall or some node the firewall is configured to trust as an intermediary in IPsec Security Parameter Index (SPI) negotiation/configuration, as that is the only visible way to demultiplex encrypted content between two the source and destination. However, even though this may mitigate the risks somewhat, but it appears that SPI's could be reused (without the intermediary) in such a way that entirely different kind of traffic could be sent. There is no fix for this, by the definition of end-to-end encryption.

A related approach could be have an intermediate firewall or security gateway act as some kind of IPsec proxy, either by formally specified means or by performing a "man-in-the-middle" -type "attack" on all the IPsec traffic. Whether this would work or be useful is not clear. A similar proposal is to require the private key storage in the security gateway; however, such an architecture would be a very attracting target and if compromised, would severely compromise the value of IPsec encryption.

One could try to encode some interesting values, e.g. protocol numbers and ports, in the Flow Label field; one problem here is the relatively limited length of 20 bits. But this would have to be done in the source node, which is not usually (at least completely) trusted in this context. Also, according to [FLOWLAB], nodes must not assume any properties in the Flow Label values.

An approach which has been proposed in the past in many forms has been to specify an IPsec-like ESP-protocol which would allow revealing only some portions of the packets, for example transport-layer headers [ESVP]. To some extent, this would be helpful, but not

necessarily enough; for example, application-specific checking might require access to the whole packet, especially if a different solution to the problem noted in the next section is not found.

A possible approach would be to try to shift the focus, at least partially, to end-node firewalls; if end-nodes are not particularly trusted, an end-node, admin-controlled firewall might be provide a reasonable tradeoff between security policy and cryptography.

There appears to be no network-based solution for this, which is indeed a feature of end-to-end cryptography.

6. Firewalls and Interactions with Peer-to-Peer Applications

6.1. The Problem

As above, the restoration of end-to-end transparency provides a possibility for a more wide-spread use of peer-to-peer applications. Such applications are often a bit problematic from the firewall perspective: it is often the practise to allow outbound (from the protected site) traffic while allowing in only the related traffic (and naturally some other administratively permitted traffic). Being able to run (some) peer-to-peer applications easily in a controlled environment would be valuable.

6.2. Possible Solutions

One workaround would be to try to standardize some default port ranges (in an application-specific manner) for such applications as these, for example in the above 32768 range. In this way, a site could enable/disable (default) port ranges for (some) peer-to-peer applications at will. A major disadvantage here would be that this could violate the trust model: some applications could intentionally try to use some other's port range to gain entry through the firewall even if the default range for that specific application was blocked. This would imply a requirement for at least some form of trust.

Another, but possibly quite a complex solution would be to implement some form of peer-to-peer "pinholing" [MIDCOM]. This hasn't yet been standardized even for IPv4 (though the concept is quite protocol-independent). A problem with model is that generally there is no trust relation between the firewall and the host (or an application at the host): how would it help if a host (or misbehaving application at the host) would be able to request opening a hole in the firewall? So, there certainly seem to be very significant tradeoffs and threat models to consider here.

A possible approach, as above, would be to try to shift, at least partially, the focus to end-node firewalls; if end-nodes are not particularly trusted, an end-node, admin-controlled firewall might be provide a reasonable tradeoff between security policy and cryptography.

7. Other Issues Associated with IPv6 Firewalls

This section tries to summarize some issues which have been brough up in conjunction with IPv6 firewalling, but which are not seen as problems as such.

7.1. IPv4 ARP vs IPv6 Neighbor Discovery

A number of people have been confused about the fact that IPv4 ARP runs at the link-layer, while IPv6 Neighbor Discovery is part of ICMPv6. When an IPv4 "deny all [IP] traffic" -rule blocks "everything except ARP", the same IPv6 rule would also deny the similar functions provided by Neighbor Discovery.

This just seems to be an issue people have to be educated on.

7.2. Filtering Specific Neighbor Discovery Messages

A typically similar set of people who have been confused of the role of Neighbor Discovery (see above) also seem to be confused on what a firewall should do with certain Neighbor Discovery packets.

It has been argued that a firewall should be able to filter out specific proxy-ND behaviour, unauthorized ND Redirects, wrong Router Advertisements, verify that packets coming from a node advertising to be reachable at some link-layer address to really come from that link-layer address, etc.

However, there are a number of strong arguments why this should not be done in a firewall. First, all of these messages are strictly on-link -- they are not routed. Thus, firewalling such messages would only be of questionable use in end-node firewalls (to protect against on-link abuse). On the other hand, as [SENDREQ] points out, the physical link is actually rather difficult to secure: in addition to enabling IP-level protections, one also has to secure link-layer -level security. Such very fine-grained, ND-specific features would seem to be clearly belong to the (Secure) Neighbor Discovery (or its implementation) itself - not the firewalls.

7.3. Firewall Policies and Multiple Addresses per Node

Often, firewall policies try to specify "a node" or "a port in a node". This naturally gets more complicated if the nodes have multiple addresses: typically at least a link-local address and a global address.

This is not problematic in itself, as the use of link-local addresses is restricted on the link (and could even be considered out of scope for most firewalls) and because one should not use link-local addresses except for specific purpose protocols. Multiple global addresses (e.g. from multihoming) can be worked out by implementation-specific methods, e.g., by making it easier to identify a node by the Interface ID part of the address when desirable, and creating the rules for all the prefixes by one pseudo-rule. However, the policies must be tuned manually if different security properties have been assigned to different prefixes.

All in all, it seems desirable to make setting policies easier also with multiple addresses, but this doesn't seem to be a problem as such.

7.4. Firewall Transparency in the Network

Many want to deploy firewalls which do not participate in the network at all, e.g. by not sending ICMPv6 unreachable packets for denied targets, but rather silently discarding any traffic they do not allow. In this kind of scenario, it may be desirable to even deploy the firewall to function as a bridge, not as a router.

Some others want to deploy firewalls to be visible: either so that the address of the firewall can be seen from the messages it sends, or so that the firewall tries to be "transparent", i.e., forge the replies as if they were coming from the destination nodes the connecting node were trying to reach (e.g., by setting the source address of an ICMPv6 message to be that of the destination address, or by forging TCP RST, etc.).

There are trade-offs both ways. A visible firewall is extremely useful when the firewall is used to set "friendly" restrictions (e.g. internally to in an Enterprise), because there will be no TCP timeouts (and similar delays) when accidentally trying something that is not allowed: an immediate ICMPv6 message or a TCP RST allows an immediate abort. The first may be useful in very hostile scenarios, where sending ICMPv6 (or other) messages might just exacerbate the issue (e.g. in the form of ICMPv6 reflection or ICMPv6 storms); however, note that ICMPv6 specification specifies rate-limiting for this specific purpose.

In any case, the firewall transparency considerations do not seem to be specific to IPv6 and are not problems as such.

8. Security Considerations

This draft discusses security considerations related to IPv6 firewalling. When discussing potential solutions for problems, the weaknesses are also pointed out.

In general, the firewall often does not, and cannot, trust the node(s) it protects. These may even belong to different administrative entit(y/ies). In that case, making compromises will usually open some holes in the firewall.

9. Acknowledgements

Brian Carpenter suggested an IPv6 firewall could support P2P pinholing. Soo Guan Eng provided commentary. Andras Kis-Szabo and Changming Liu provided a number of comments and useful commentary.

10. References

10.1. Normative References

[ADDRARCH] Hinden, R., Deering, S., "IP Version 6 Addressing Architecture", RFC3513, April 2003.

[IPV6] Deering, S., Hinden, R., "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

10.2. Informative References

[DISFW] Bellovin, S., "Distributed Firewalls", ;login: in Nov 1999, http://www.research.att.com/~smb/papers/distfw.html

[ECN] Garzik, J., "ECN-under-Linux Unofficial Vendor Support Page", http://gtf.org/garzik/ecn/, March 2002.

[ESVP] Kasera, S. (ed.), "IP Encapsulating Security Variable Payload (ESVP)", work-in-progress, draft-kasera-esvp-00.txt, October 2002.

[FLOWLAB] Rajahalme, J., et al., "IPv6 Flow Label Specification", work-in-progress, draft-ietf-ipv6-flow-label-07.txt, April 2002.

[ICMPV6] Conta, A., Deering, S., "Internet Control Message Protocol (ICMPv6)", RFC2463, December 1998.

[MIDCOM] Srisuresh, P. et al, "Middlebox communication architecture and framework", RFC3303, August 2002.

[MIPV6] Johnson, D., et al, "Mobility Support in IPv6", draft-ietf-mobileip-ipv6-24.txt, work-in-progress, July 2003.

[RFC1122] Braden, R. (Editor), "Requirements for Internet Hosts -- Communication Layers", RFC1122, October 1989.

[RFC1812] Baker, F. (Editor), "Requirements for IP Version 4 Routers", RFC1812, June 1995.

[RHHAOSEC] Savola, P. "Security of IPv6 Routing Header and Home Address Options", work-in-progress, draft-savola-ipv6-rh-ha-security-03.txt, December 2002.

[SENDREQ] Nikander, P., el al., "IPv6 Neighbor Discovery trust models and threats", work-in-progress, draft-ietf-send-psreq-03.txt, April 2003.

Author's Address

Pekka Savola
CSC/FUNET
Espoo, Finland
EMail: psavola@funet.fi

A. Possible Desirable Header Field Matching Extensions

As Destination options and Extension Header types are taken into use, it may be desirable for a firewall to support some matching against certain header fields. These include, for example:

- whether or not a specific Extension Header or a Destination Option is detected
- behaviour when an unknown (or specified) Extension Header or Destination Option is detected
- (Routing Header -specific) being able to match segments left (mainly, whether it is zero or not), type and the next-to-be-swapped destination(s) [RHHAOSEC]
- (Home Address Option [MIPV6] -specific) being able to match against the home address
- (ESP/AH -specific) being able to match against SPI
- (Tunneled-traffic specific) being able to match against the embedded IPv4 address in e.g. 6to4, ISATAP, etc.

Some of these are much more useful than others; the list is only meant to give ideas about possibly useful (in some scenarios, at least) functionalities.

B. Amplification DoS Attack Using IPv6 Multicast

It is possible to launch a denial-of-service attack using IPv6, including a form of amplification based on multicast infrastructure.

Multicast address must not be used as a source address ([ADDRARCH], section 2.7), but explicit checks to drop those packets or respond to them have not been specified (that is, [ADDRARCH] specifies that nodes MUST discard certain kinds of packets if received, but these are not listed as such). However, such attacks are not considered here.

By crafting packets, sent to multicast destinations, which are certain to contain a response to the source address (the victim's address) would seem to be potentially useful for an attacker:

```
src = <victim> (unicast-address)
dst = <multicast address> (w/ as many receivers as possible)

next-header=200 (something undefined)
  or:
next-header=destination-options (or hop-by-hop options)
options-header=<an unknown option type starting with binary "10">
```

Now, both of these amount to the same thing: every node which processes the packet and adheres to [IPV6] will generate an ICMPv6 parameter problem back to the source.

[ICMPV6] does not permit the first approach with an unknown extension header, but additionally permits ICMPv6 message generation with Path MTU Discovery.

This is different with IPv4, where no ICMP errors are generated in response to packets sent to multicast addresses [RFC1122]. Clearly, when specifying, allowing the flexibility to define whether a response to multicast packets would be sent was considered a feature, but dangerous features have a tendency to being turned to bad uses.

In addition to amplification, this kind of attack would have an effect on multicast-enabled router network as a large amount of multicast forwarding state would be created.

Intellectual Property Statement

Full Copyright Statement

MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.