Covert Channels in IPv6

Norka B. Lucena, Grzegorz Lewandowski, and Steve J. Chapin

Syracuse University, Syracuse NY 13244, USA
{norka, chapin}@ecs.syr.edu, grlewand@syr.edu

Abstract. A covert channel is a communication path that allows transferring information in a way that violates a system security policy. Because of their concealed nature, detecting and preventing covert channels are obligatory security practices. In this paper, we present an examination of network storage channels in the Internet Protocol version 6 (IPv6). We introduce and analyze 22 different covert channels. In the appendix, we define three types of active wardens, *stateless*, *stateful*, and *network-aware*, who differ in complexity and ability to block the analyzed covert channels.

Keywords: covert channel, IPv6, active warden, stateless, stateful, IPsec.

1 Introduction

When analyzing the security of computer systems, it is important to evaluate both overt and covert communication channels. An *overt* channel is a communication path within a computer system or network designed for the authorized transfer of data. Authorized data transmission involves the existence of security policies as well as mandatory and discretionary access controls that restrict the flow of information. A *covert* channel, in contrast, is a communication path that allows an unauthorized process to transfer information in a way that violates a system security policy [1]. Because of the concealed nature of covert channels, detecting and preventing them are obligatory practices in multilevel security systems (MLS) where most of the processes carry classified information [2].

Covert channels are primarily classified as storage or timing channels. A *storage* channel concerns “the direct or indirect writing of a storage location by one process and the direct or indirect reading of it by another” [3]. A *timing* channel involves signaling mechanisms based on modulation of system resources such as CPU or time in a way that the change in response time observed by a second process conveys information. This paper focuses in the study of covert storage channels in the Internet Protocol version 6 (IPv6) (also called the Next Generation Internet Protocol or IPng). This initial examination is a specification-based analysis, to identify redundancies and ambiguities in the protocol semantics that could potentially be used to carry covert data.

Network-based covert channels can be used both as a means of private communication and as a means to coordinate distributed denial of service or other kinds of attacks [4]. In addition, hackers favor network storage channels over timing channels because of the synchronization issues and significantly lower bandwidth of the latter. Our research aims to generate discussion of such channels, and also to raise issues for consideration by implementors of IPv6 protocol stacks and firewalls that handle IPv6 traffic.

The remainder of this document is organized as follows. Section 2 summarizes the work done in network covert channels detailing the findings corresponding to storage channels. Section 3 presents the adversary model under which the existence of covert channels will be analyzed. Section 4 discusses potential covert channels in IPv6 as well as issues regarding security.Finally, section 5 draws some conclusions and points out directions for future work. Appendix A analyzes the defined covert channels in a network monitored by several types of active wardens.

2 Related Work

Previous research in network covert channels [4] focuses on Internet Protocol version 4 as well as other related protocols such as TCP, ICMP, and HTTP[4,5,6,7,8,9,10,11,12, 13,14,15,16,17,18,19,20] . The study of network storage channels[9,10,11,12,13,14,15, 16] is broader than its counterpart of network timing channels [4, 5, 6, 7, 8], presumably because of the synchronization issues present in timing channels and their low bandwidth in comparison to storage channels. This work aims to analyze storage channels in the new generation of the Internet Protocol, IPv6.

Handel and Sandford [9] pioneers covert channels within network communication protocols. It describes different methods of creating and exploiting hidden channels in the OSI network model, based on the characteristics of each layer. Szczypiorski [10] describes a hidden communication system at the data link layer of the OSI network mode that takes advantage of imperfections in the transmission medium, such as interferences and noise. Rowland [11], Dunigan [12], and Rutkowska [13] present examples of implementation of covert channels that exploit header fields of the TCP/IP protocol suite (for IPv4). These three papers focus their attention in the network and transport layers of the OSI network model.

Abad [14] describes how to embed data in the IP checksum using selected hash collisions. The IPv4 checksum can be exploited because the algorithm used to calculate it is susceptible to collision attacks. In IPv6, checksums are calculated by keyed message authentications codes (MAC) based on symmetric encryption algorithms such as DES or on one-way hash functions such as MD5 or SHA-1. One-way hash algorithms will reduce, but probably not eliminate because of recent MD5 collisions [15, 16], the possibility of existence of similar channels in IPv6.

Giffin et al. [17] analyzes a low-bandwidth covert channel that uses TCP timestamps. The channel is based on a modification of a TCP header field, in particular, the low order bit of the timestamp option. In a slow connection, this channel is harder to detect than the ones described in [12, 11] because under such network conditions the low order bit of the timestamp appears randomly distributed facilitating the transmission of encrypted messages.

Ahsan and Kundur [5, 6] proposes five covert channel approaches: four of them based on manipulations of the TCP, IGMP, and ICMP protocol headers and one of them based on packet sorting within the IPsec protocol. The former are storage channels while the latter is a timing channel. The network timing channel works by sorting packets by the sequence number field present in both the authentication header (AH) and the encapsulated security payload header (ESP) defined in IPsec. The hidden information is the difference between the original sequence of packets and the sorted sequence.

Project Loki[1] [18, 19] explores the concept of ICMP tunneling, exploiting covert channels through the data portions of the ICMP_ECHO and ICMP_ECHOREPLY packets. The Loki client allows a remote attacker to wrap and transmit commands in ICMP payloads. Lokid, the Loki server, unwraps and executes the commands, sending the results back wrapped in ICMP packets. Back Orifice 2000 with the BOSOCK32 plug-in also implements covert channels via ICMP. Firewalls can disallow entirely the passing of ICMP traffic, preventing the existence of tunneling. Project Loki also runs over UDP on port 53, simulating DNS traffic. Sneakin [20] provides an incoming shell through outgoing Telnet-like traffic.

Currently, the most effective defensive mechanisms against network storage channels for IPv4 are protocol scrubbers [21], traffic normalizers [22], and active wardens [23]. Both protocol scrubbers and traffic normalizers focus on eliminating ambiguities found in the traffic stream presumably created by a skilled attacker with the purpose of evading network intrusion detection systems. Fisk et al. [23] defines two classes of information in network protocols: *structured* and *unstructured* carriers. Structured carriers present well-defined, objective semantics, and can be checked for fidelity en route (e.g., TCP packets can be checked to ensure they are semantically correct according to the protocol). Unstructured carriers, such as images, audio, or natural language, lack objectively defined semantics and are mostly interpreted by humans rather than computers. Analyzing the Linux and OpenBSD implementation of the TCP/IP protocol stack, Murdoch and Lewis [24] shows that fields commonly used for steganography, such as the IP identification field and the TCP initial sequence number, exhibit enough structure and nonuniformity to facilitate detection.

3 Adversary Model

In the context of the "classical" prisoners' problem [25], *Alice* and *Bob* are two agents who wish to communicate covertly (see Figure 1). As described in [26], Alice and Bob exploit an already existing communication path, corresponding to two arbitrary communicating processes: the *sender* and the *receiver*. *Wendy* is a warden, located somewhere along the communication path, monitoring all possible messages exchanged by Alice and Bob.

The dotted boxes in Figure 1 indicate that Alice and Bob could either act as sender and receiver, or could modify the messages in transit [26]. From Wendy's point-of-view, these situations are indistinguishable.

In this framework, Wendy always acts as an *active* warden [23, 27, 28]. Active wardens can modify the content of the network traffic with the purpose of eliminating any form of hidden communication. When modifying network packets, active wardens should maintain the syntactic and semantic integrity of the packet to avoid breaking the overt communication. They reinforce protocol specifications through mechanisms such as zeroing reserved fields, randomizing ID numbers, and requiring or prohibiting the use of option fields.

[1] Loki is pronounced "low-key", and is named for the Norse god of trickery.

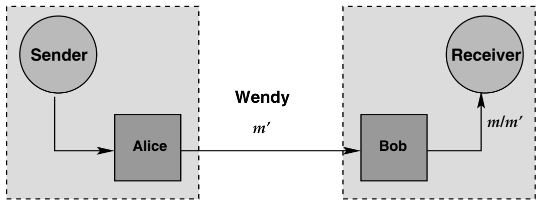


Fig. 1. Framework for Covert Communication

4 Potential IPv6 Covert Channels

The IPv6 header structure has a fixed length of 40 bytes. Five fields from IPv4 were removed (header length, identification, flags, fragment offset, and header checksum). Options are defined as extension headers. A packet can have more than one extension header. When present, the headers are layered in order. The IPv6 protocol specification, RFC 2460 [29], defines six extension headers [30]:

- Hop-by-Hop Options header
- Routing header
- Fragment header
- Destination Options header
- Authentication header (AH)
- Encapsulating Security Payload (ESP) header

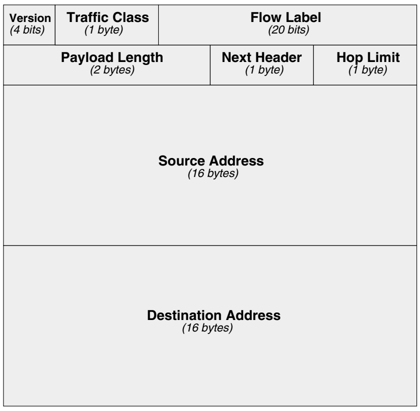
The last two extension headers and their functionality are described in separate RFCs as part of the IP security framework (IPsec): RFC 2401 [31], RFC 2402 [32], and RFC 2406 [33]. The security architecture for the Internet protocol, RFC 2401, establishes that the AH and ESP header can be used in two modes: transport mode and tunnel mode. In *transport* mode, encryption or authentication are applied to the payload contained in all IP packets related to particular end-to-end connection. In *tunnel* mode, authentication and encryption mechanisms are defined between two security gateways[2], surrounding both the IP header and the payload with a "wrapper" IP packet.

Our analysis of potential covert channels in IPv6 includes a specification-based covert channel discovery and an informal bandwidth estimation. The presence of AH and ESP headers in any of the modes affects some of the presented covert channels. The examination also points out such effects. Covert channels are described by header and identified with a letter from the Greek alphabet.

4.1 IPv6 Header

Figure 2 shows the fields in the IPv6 header as well as the plausible covert channels observed.

[2] An intermediate system that implements the IPsec framework, e.g. a firewall implementing IPsec.



ID	Field	Covert Channel	Bandwidth
α	*Traffic Class* class	Set a false traffic	8 bits/packet
β	*Flow Label* label	Set a false flow	20 bits/packet
γ	*Payload Length*	Increase value to insert extra data[3]	Varies
δ	*Next Header*	Set a valid value to add an extra extension header[3]	Varies
ϵ	*Hop Limit*	Increase/decrease value	$\approx$ 1 bit/packet
ζ	*Source Address*	Set a false source address	16 bytes/packet

(a) (b)

Fig. 2. Covert Channels in the IPv6 Header. (a) IPv6 Header Format, (b) Identified covert storage channels.

α Alice can set a false *traffic class* value. The bandwidth of this channel varies up to 8 bits per packet, depending on whether or not the field is modified by intermediate nodes. The IPv6 specification allows the intermediate nodes to change the value of the traffic class field as they forward the packet. For example, Differentiated Services traffic conditioner [34] might modify the traffic that passes through it. Therefore, when Alice and Bob communicate using this covert channel, they have to be prepared to handle disturbances. Additionally, intermediate nodes might use this field to make decisions about packet processing, thus covert channel that manipulates the value of this field might have an unpredictable effect on the network.

β Fabricating a *flow label*, Alice can send 20 bits of data per packet. Authentic flow labels are pseudo-randomly and uniformly selected numbers, ranging from 1 to FFFFF hex. Alice needs to preserve the same conditions when creating a fake flow label.

γ Alice can increase the value of the *payload length* and append extra data at the end of the packet. The bandwidth of this channel varies depending on the size of the original packet, but the modified packet cannot be larger than 65536 bytes. If encryption is used without authentication, stego techniques like the ones described in [26] are appropriate. If authentication is used, Alice and Bob need to take extra steps to maintain the covertness of the channel because the payload length is included in the calculation of the integrity check value (ICV). The ICV is a field of the Authentication Extension Header calculated over several fields from the IP header and from the extension headers, when present, used to verify whether a packet was corrupted or modified in transit. See subsection 4.6 for details.

δ Because extension headers are not examined nor processed by intermediate nodes of an end-to-end communication path, Alice can change the *next header* content

[3] This covert channel, when authentication is used, requires recalculating or circumventing the integrity check value (ICV). See subsection 4.6 for details.

to insert an entire extension header covertly. This channel will, obviously, require that Alice increases the payload length accordingly. The bandwidth of this channel depends on the total length of the extension header inserted. An end-point node that does not recognize the value in the *next header* field[4] will discard the packet and send an ICMP notification to the source. Alice and Bob could also use the ICMP reply as a means of covert communication.

ϵ Alice can initiate a covert communication channel by setting an initial *hop limit* value, h, and manipulating the *hop limit* value of subsequent packets. Bob interprets the covert message by checking the variations in the hop limit values of packets traversing his location. One scheme has Alice signaling a 0 by decreasing the hop count from the prior packet, and a 1 by increasing the hop count relative to the prior packet. A drawback of this channel is that packets do not necessarily travel the same route, so the number of intermediate hops may vary, introducing noise. To overcome this, Alice can choose a δ that is greater than the expected noise, and use hop counts less than $h - \delta$ signal a 0, and hop counts greater than $h + \delta$ to signal a 1. Bob then compares the received hop count to h to deduce the bit. The bandwidth of this channel is limited. Alice needs to modify n packets to send $n - 1$ bits of information.

ζ Alice can forge the source address field to send 16 bytes of covert data. Detection of this channel is however very likely because of the existing security mechanisms that detect source address spoofing.

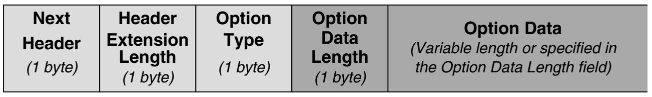
4.2 Hop-By-Hop Options Header

The hop-by-hop options header carries optional information that needs to be checked by every node the packet traverses. Because of its different option types, both defined and undefined, and its variable length, this extension header offers possibilities for high-bandwidth covert channels. As described in the protocol specification [29], the *option type* field is an octet structure that has three subfields: the first two bits specify what action should be taken when an unrecognized option is received; the next bit determines whether or not the option data can change in route; the last five bits represent the option number[5]. The analysis introduced below discusses relevant types of option such as the padding, jumbogram, and routing alert options (see Figure 3). When authentication is used, most of the covert channels in the hop-by-hop header may require recalculating or circumventing the ICV (see discussion in subsection 4.6).

α Jumbograms are IPv6 packets with payload length longer than 65535 bytes. Alice can use jumbograms as a means of covert communication in two ways. The first one relies on modifying an existing jumbogram length with the purpose of appending covert data (this mechanism relates to the δ channel is subsection 4.1). The second method involves converting a regular datagram into a jumbogram and filling in the extra bytes with hidden content. It will be necessary, consequently, to change the payload length in the IPv6 header. Jumbograms are discarded by intermediate

[4] The Protocol Numbers document [35] lists of all possible *next header* field values.

[5] These last five digits are also called “option type” or “rest”. However, the option type is fully specified only when using the entire octet.


Next Header (1 byte)
Header Extension Length (1 byte)
Option Type (1 byte)
Option Data Length (1 byte)
Option Data (Variable length or specified in the Option Data Length field)

(a)

ID	Field	Covert Channel	Bandwidth
α	*Option Type: Jumbogram*	Insert or create a jumbogram	Varies
β	*Option Type: Router Alert*	Set a false router alert	2 bytes/packet
γ	*Option Type: PadN*	Set a false padding value	Up to 256 bytes/packet
δ	*Option Type: Unknown*	Fabricate one or more options	Up to 2038 bytes/packet

(b)

Fig. 3. Covert Channels in the Hop-by-Hop Options Extension Header. (a) Format of the Hop-by-Hop Options Header, (b) Identified covert storage channels.

nodes that do not support them. Therefore, Alice and Bob need to make sure that all nodes in the communication path understand jumbograms.

β Router alert options contain a 2-byte *reserved* field where Alice can embed data to establish a covert communication. Alice could also add an entire router alert option type, if it does not exist. That alternative will require readjustment of the packet length in the IPv6 header.

γ Individual options in the option data field need to preserve header alignment. Two types of padding are defined for that: Pad1 and Pad$N$. Pad1 inserts a single octet, Pad$N$ appends two or more bytes as an individual option type. Alice can exploit any of the padding types, but γ focuses only in the Pad$N$ *option type*. A simple form of using this option is to embed covert data in an already-existing padding. The bandwidth of that channel will depend then in the length of the padding option. A more crafted way would be inserting a padding option when the header does not contain one. Alice could send this way up top 256 bytes/packet because the Pad$N$ option has a maximum length of 256 bytes. The last alternative, illustrated in Figure 4, requires modification of the IPv6 payload length.

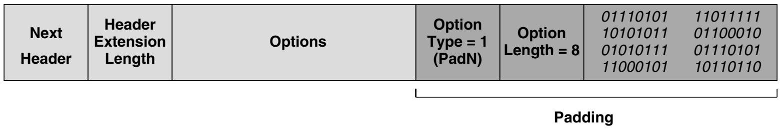
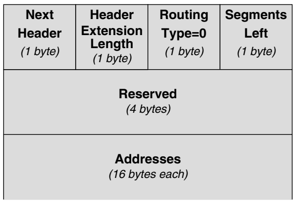


Fig. 4. δ **Covert Channel in the Hop-by-Hop Options Extension Header**

δ Alice can fabricate an option type, different from the ones listed in [36], as long as she maintain the semantics of the field described at the beginning of the subsection. She needs to make the first two bits of the *option type* equal to 00. That will dictate intermediate nodes to “skip and continue processing” when they do not recognize the option type [29]. The maximum length of option data is 256 bytes. Therefore,



(a)

ID	Field	Covert Channel	Bandwidth
α	*Routing Type: 0 - Reserved*	Hide data in unused bits	4 bytes/ packet
β	*Routing Type: 0*	Set one or more false addresses[7]	Up to 2048 bytes/packet

(b)

Fig. 5. Covert Channels in the Routing Extension Header. (a) Format of the Routing Header, (b) Identified covert storage channels.

up to 256 bytes of covert data can be inserted that way. Moreover, because the hop-by-hop header can include many options, by repeating the insertion with different option type values, up to 2,038 bytes can be added in total[6]. Inserting new options increases the total length of the IPv6 packet.

4.3 Routing Header

The routing extension header contains a list of intermediate nodes a packet in transit should visit on the way to its destination. The IPv6 Parameters document [36] enumerates three different types of routing, but only one of them, *Type 0*, is fully described in the specification [29]. Figure 5 shows the format of the routing header when routing type is 0 and its possible channels.

α There exists a *reserved* field in routing header structure when the *routing type* is 0. Alice can hide 4 bytes of covert data per packet using this channel.

β When the *routing type* is 0, Alice can fabricate "addresses" out of arbitrary data meaningful to Bob. She appends the covert data and sets the *segments left* field to 0 to prevent any node to attempt processing the fake addresses. Figures 6 and 7 display two different types of embedding:
- one where Alice chooses to create a new routing extension header of routing type 0 to send Bob 48 bytes of covert information
- another one where she takes advantage of an already existing routing extension header of routing type 0 to embed a covert message of 32 bytes.

Based on the maximum extension header payload length, Alice can potentially insert up 2048 bytes. Therefore, she will be extending the entire IPv6 packet by the same amount of bytes.

4.4 Fragment Header

As in IPv4, fragmentation of packets occurs when the MTU of a link is not large enough to handle a packet of a particular size. Unlike IPv4, IPv6 packets are not fragmented by

[6] The length of the header payload is 2054 bytes, which can be filled by 7 options carrying 256 bytes each and 1 option of 246 bytes considering that the headers of individual options will require 16 bytes.

[7] This covert channel, when authentication is used, requires recalculating or circumventing the ICV. See subsection 4.6 for details.

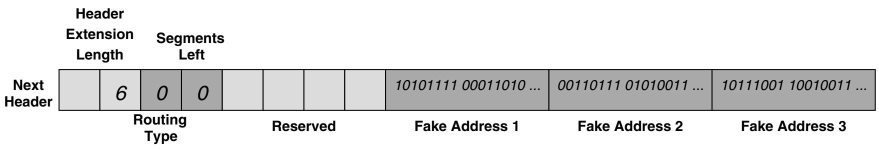


Fig. 6. α. **Covert Channel in the Routing Extension Header.** When Alice creates fake addresses in a packet that did not originally a routing extension header.

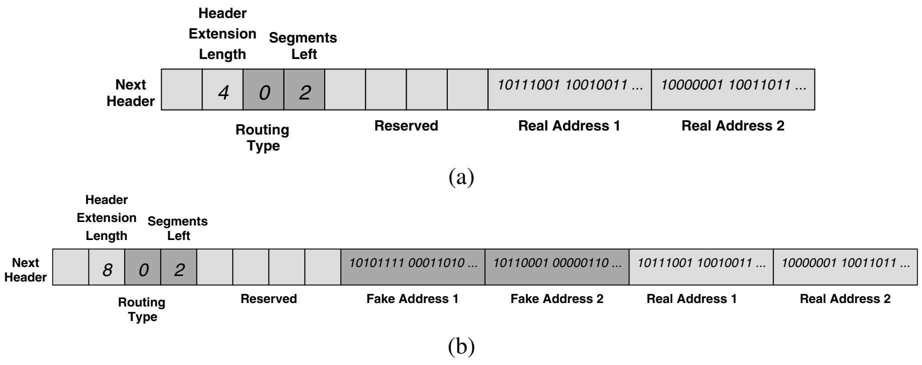


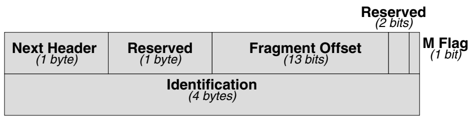
Fig. 7. α. **Covert Channel in the Routing Extension Header.** When Alice inserts fake addresses in a packet already containing a routing extension header. (a) Original routing extension header, (b) Routing header after Alice inserts the covert data.

routers along the path. Sending hosts use path MTU discovery to determine the allowed maximum packet size on the way to a specific destination. Sending hosts fragment packets accordingly, when necessary. Destination hosts reassemble them.

An important consideration regarding fragmented packets is that they are themselves IPv6 packets, thus all previously described covert channels exist in fragments as well. In addition, because the number of packet fragments is presumably greater than the original number of packets to be sent by a host, the opportunities for information hiding also increase. On the other hand, new covert channels appear when a large packet is fragmented. Channel γ is an example of such case.

Figure 8 displays the format of the Fragment Extension Header and its potential covert channels.

- α Alice can transmit 8 bits of covert data using the first *reserved* field of the header. This field is initialized to zero by the sending host, but it is ignored by the destination. Therefore, at the receiving end a value different from zero makes no difference.
- β 2-bit *reserved* field has the same treatment as the 8-bit reserved field, so Alice can exploit it taking a similar approach.
- γ The reassembly process at the destination host takes into account only the *next header* value of the first fragment as a reference. Also, it ignores the *next header* values of fragments that differ. Those conditions give Alice the opportunity to em-



(a)

ID	Field	Covert Channel	Bandwidth
α	*Reserved*	Hide data in the unused bits	8 bits/packet
β	*Reserved*	Hide data in the unused bits	2 bits/packet
γ	*Next Header*	Set a false next header	At least 8 bits/fragment
δ	*All*[8]	Insert an entire fake fragment	Up to 64 KB/ fragment

(b)

Fig. 8. Covert Channels in the Fragment Extension Header. (a) Format of the Fragment Header, (b) Identified covert storage channels.

bed 8 bits of covert data per fragment as long as she keeps the next header value of the first fragment untouched.

δ Alice can potentially insert an entire fragment exploiting *all* fields of the fragment header. To avoid having this fragment included in the reassembly of the original packet, she can assign an invalid fragment ID field, so that the receiver will discard it. The bandwidth of this channel depends on the size of the fragment. Figure 9 shows a graphical representation of this channel.

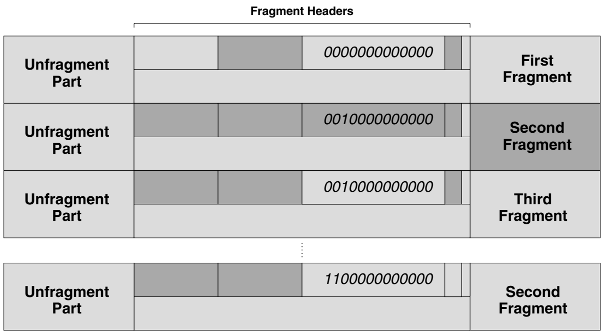


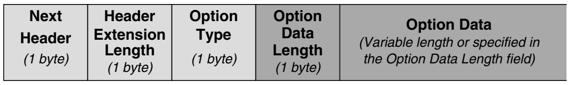
Fig. 9. δ Covert Channel in the Fragment Extension Header. Alice inserts a fake fragment in the fragments stack, setting a *fragment offset* value that causes its data to be overwritten in reassembly.

4.5 Destination Options Header

The Destination Options Header carries optional information only relevant to the destination nodes. It may appear twice in the IPv6 headers stack: a) after the hop-by-hop header when has options that need to be processed by the first destination in the IPv6 header and the ones listed in the router header; and b) after all other extension headers when it carries options to be processed only by the final destination.

[8] This covert channel, when authentication is used, requires recalculating or circumventing the ICV. See subsection 4.6 for details.

Because options of both extension headers, hop-by-hop and destination, follow the same option format, the covert channels identified are similar to those shown in Figure 10. Details of how to exploit those channels are described in subsection 4.2. In addition, the Swiss Unix User Group reports an implementation of the covert channel α [37].


Next Header (1 byte)
Header Extension Length (1 byte)
Option Type (1 byte)
Option Data Length (1 byte)
Option Data (Variable length or specified in the Option Data Length field)

(a)

ID	Field	Covert Channel	Bandwidth
α	*Option Type: Unknown*	Fabricate one or more options[9]	Up to 2038 bytes/packet
β	*Option Data: Padding*	Set a false padding value[10]	Up to 256 bytes/packet

(b)

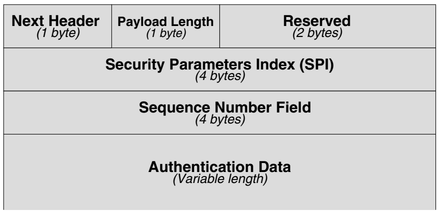
Fig. 10. Covert Channels in the Destination Options Extension Header. (a) Format of the Destination Options Header, (b) Identified covert storage channels.

4.6 Authentication Header

The Authentication Extension Header (AH) is the one of the two headers that compose IPsec. It provides connectionless integrity and data origin authentication of individual IP packets. It does so by calculating an integrity check value (ICV) per packet based on particular fields from other extension headers and from the IPv6 header as well. Whether a header field is actually used in the ICV computation or not depends on its mutability in transit. Only fields whose values do not change or change in a predictable way along the communication path are included in the computation. Other fields that may vary en-route, such as the *option data* field in options headers, are set to zero before being included calculation to avoid modifications in length or alignment. If a covert channel technique involves modifying a *immutable* or *mutable predictably* header field protected by authentication, Alice and Bob need to take special actions so their covert communication is not broken. This subsection discusses both potential covert channels in the authentication header and possible solutions the agents can apply when using previously discussed channels over authenticated headers. Figure 11 shows the structure of the authentication header and its potential covert channels.

- α This channel is similar to the channel α from subsection 4.4. It has however twice as much bandwidth.
- β When the authentication header is not present, Alice can fabricate one and insert it in the stack of extension headers. Alice has to set appropriate values for the *next header*, *payload length*, *security parameters index*, and *sequence number* to avoid detection. She places the covert data in the field that apparently contains *authentication data*. Obviously, the fake authentication header will not pass the IPsec integrity

[9] These channels involve changing the packet total length, which affects the ICV, when authentication is present. Subsection 4.6 describes that situation.



(a)

ID	Field	Covert Channel	Bandwidth
α	*Reserved*	Hide data in the unused bits	2 bytes/packet
β	*All*	Insert an entire fake header	Up to 1022 bytes/packet

(b)

Fig. 11. Covert Channels in the Authentication Extension Header. (a) Format of the Authentication Header, (b) Identified covert storage channels.

check at the receiving end. Therefore, Bob needs to strip it before the packet authentication check. Alice can send Bob up to 1022 bytes per packet through this channel (see Figure 12) Notice that this channel also involves modifying the size of the original packet, but this time the *payload length* in the IPv6 is not actually authenticated because there is no real AH.

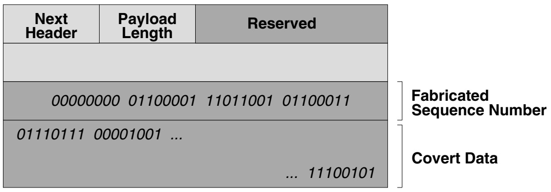


Fig. 12. β **Covert Channel in the Authentication Header.** Alice inserts fake authentication header in the stack of headers, simulating a *sequence number* to defeat active wardens.

Issues with the Authentication Header Integrity Check Value (ICV): The ICV computation consists of applying message authentication code (MAC) algorithms over immutable and mutable but predictable fields from the IPv6 header and its extension headers. Several of the proposed covert channels involve changing values of some of those protected fields. The existence of this channel can cause failure of the integrity check, which triggers an auditable event in IPv6. That may cause both immediate detection of the channel and disruption of the overt communication. Alice and Bob must take actions to avoid such situations. The following, Table 1, table summarizes the affected covert channels and notes the nature of the field protected by the ICV.

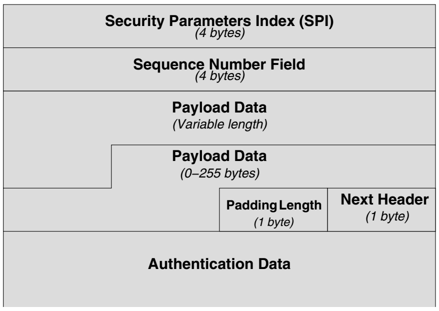
To avoid a failed check on the ICV, Alice must either be the sender, and therefore compute the ICV including the covert data, or Bob must intercept the packet before it reaches its destination, and remove the covert data, as described in [29].

4.7 Encapsulating Security Payload Header

Also part of IPsec, the Encapsulating Security Payload (ESP) Header provides confidentiality for all data transmitted end-to-end in IP packets. The general structure of the ESP header and its plausible covert channels are illustrated in Figure 13.

Table 1. Covert Channels Affected by the ICV Calculation

Affected Covert Channel	Protected Field
γ channel in 4.1	*Payload Length* (immutable)
δ channel in 4.1	*Next Header* (immutable)
α, β, γ, and δ channels in 4.2, when changing packet size	*Payload Length* (immutable)
α 4.3, when changing the size of original packet	*Payload Length* (immutable)
δ 4.4, when changing the size of original packet	*Payload Length* (immutable)
α and β channels in 4.5, when changing packet size	*Payload Length* (immutable)



(a)

ID	Field	Covert Channel	Bandwidth
α	*Padding*	Set a false padding value	Up to 255 bytes/packet
β	*All*	Insert an entire fake header	Up to 1022 bytes/packet

(b)

Fig. 13. Covert Channels in the Encapsulating Security Payload (ESP) Extension Header. (a) Format of the ESP Header, (b) Identified covert storage channels.

α Although the *padding* field in the ESP header is optional, all IPv6 must support them. Alice can send up to 255 bytes per packet exploiting this channel.

β When the ESP header is not present, Alice can fabricate an entire ESP-like header to transmit covert information. Because the ESP header is an encapsulating header, she will need to include the original payload when creating her own. As in channel β of the AH, a fake ESP header will not pass through the IPsec verification. Therefore, Bob needs to remove it, restoring the packet to its original form, before the packet reaches the final destination. Figure 14 shows an example of this channel.

Effects of the Encapsulated Security Payload Header: As shown in Figure 13(a), the ESP header includes an authentication field. However, the ESP integrity check applies only to the ESP internal fields, the encapsulated headers, and the payload. That implies that, in transport mode, the presence of the ESP header does not affect the covert channels previously described, with exception of the ones belonging to the destination options header because that header is placed after the ESP header (i.e., it is encapsulated). To exploit the destination options header channels, Alice and Bob need access to the encryption keys. In tunnel mode, the "inner" IP header and all its extensions are encapsulated from source to destination in the "outer" IP header. However, ESP tunnels can still be used for secret communication if Alice piggybacks an encrypted covert message to the "outer" header payload [29].

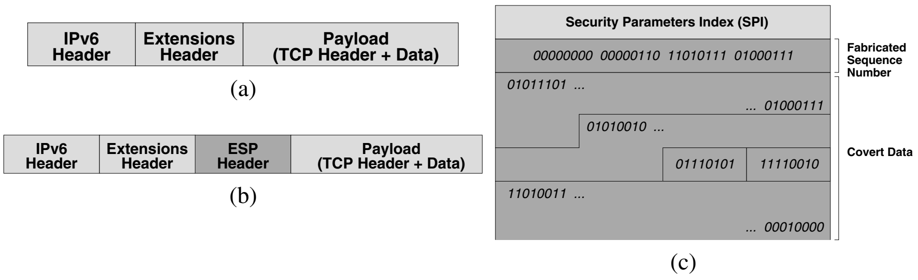


Fig. 14. **β Covert Channel in the Encapsulating Security Payload (ESP) Extension Header.** (a) Packet before inserting the fake ESP header, (b) Packet after insertion, (c) Detail of the fabricated header.

4.8 Covert Channels in Tunneled Traffic

The use of IPsec authentication and encryption in tunnel mode affects the covert communication. The implications are primarily related to the location of the agents communicating covertly. In the presence of tunneled traffic, Alice and Bob need to locate themselves in particular sprts along the communication path. There are three possible tunnel configurations based on the interaction of different versions of the Internet protocol:

IPv6 Traffic in an IPv6 Tunnel: In this scenario both inner and outer headers follow the IPv6 specification, hence both can carry covert data using the techniques in the described channel. Moreover, both headers provide two independent covers for hiding information.

- An **authenticated tunnel** affects the embedding and extraction of covert data in both inner and outer headers.
  When Alice and Bob are outside the tunnel (see Figure 15), they communicate covertly by modifying field within the inner headers. If Wendy is within the tunnel as in Figure 15(a), any countermeasures might cause a failure in the authentication check, breaking the overt communication as well. On the other hand, if Alice and Bob are within the tunnel, as in Figure 15(b), they most likely modify the outer header. Wendy can block their channel without major implications.
- An **encrypted tunnel** alters the embedding and extraction of data hidden in the inner headers. Because inner headers are encrypted, any agent who wishes to modify them must either have knowledge of the encryption key or hide the data before the tunnel is applied.

IPv6 Traffic in an IPv4 Tunnel: This is the most common case present today. Because our interest is only IPv6, the concern here is with the covert channels present in the inner header. However, the IPv4 tunnel interferes with all agents trying to monitor or modify the inner IPv6 traffic. If the IPv4 tunnel does not use IPsec, Alice, Bob, and Wendy just need to understand both IPv4 and IPv6 headers. If the IPv4 tunnel employs IPsec, the encountered problems are exactly as discussed in the previous scenario.

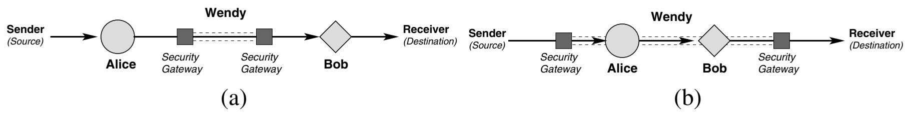


Fig. 15. Location of Alice, Bob, and Wendy under and IPsec Tunneling Mode. (a) Alice and Bob embed and extract, respectively, covert data outside the tunnel; Wendy is within the tunnel, so an attempt to modify the traffic might cause the authentication to fail. (b) Alice, Bob, and Wendy are all inside the tunnel; the warden can prevent covert channels in the outer header.

IPv4 Traffic in an IPv6 Tunnel: In this case, the IPv4 traffic is treated as any other payload within a stream of IPv6 packets.

5 Conclusions

In this paper we have presented a comprehensive overview of covert channels existing in the Internet Protocol version 6. We have found and analyzed 22 covert channels.

We hope that this analysis is useful to implementors of firewalls that understand IPv6 traffic, attracting their attention towards harmful covert channels at the IP level. Especially, considering the recent trend of deep packet inspection firewalls, which evolve towards the application layer.

Our future work will include an analysis of covert channels within the Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6, implementation of a software package that demonstrates communication using the covert channels described here, more formal calculation of bandwidth available in the covert channels, statistical analysis of IPv6 traffic carrying covert communication.

Appendix A defines three types of active wardens, *stateless*, *stateful*, and *network-aware*, which differ in complexity and ability to block the covert channels introduced in this paper. Additionally, we discuss the countermeasures active wardens can undertake to detect and defeat those channels.

References

1. Gligor, V.D.: A Guide to Understanding Covert Channel Analysis of Trusted Systems. National Computer Security Center, Meade, MD, USA. Version-1 edn. (1993) NCSC-TG-030.
2. McHugh, J.: Covert Channel Analysis: A Chapter of the Handbook for the Computer Security Certification of Trusted Systems. Portland State University, Portland, Oregon, USA. (1995)
3. of Defense, U.D.: Department of Defense Trusted Computer System Evaluation Criteria. (1985) DOD 5200.28-STD.
4. Cabuk, S., Brodley, C.E., Shields, C.: Ip covert timing channels: Design and detection. In: Proceedings of the 11^{th} ACM Conference on Computer and Communications Security, Washington DC, USA, ACM Press (2004) 178–187
5. Ahsan, K.: Covert channel analysis and data hiding in tcp/ip. Master's thesis, University of Toronto (2002)
6. Ahsan, K., Kundur, D.: Practical data hiding in tcp/ip. In: Proceedings of the ACM Workshop on Multimedia Security at ACM Multimedia. (2002)

7. Servetto, S.D., Vetterli, M.: Codes for the fold-sum channel. In: Proceedings of the 35^{35} Annual Conference on Information Science and Systems (CISS), Baltimore, MD, USA (2001)
8. Servetto, S.D., Vetterli, M.: Communication using phantoms: Covert channels in the internet. In: Proceedings of the IEEE International Symposium on Information Theory (ISIT), Washington, DC, USA (2001)
9. Handel, T., Sandford, M.: Hiding data in the OSI network model. In Anderson, R., ed.: Information Hiding: Proceedings of the First International Workshop. Volume 1174., Cambridge, U.K., Springer (1996) 23–38
10. Szczypiorski, K.: Hiccups: Hidden communication system for coruppted networks. In: Proceedings of the Tenth International Multi-Conference on Advanced Computer Systems ACS'2003, Międzyzdroje, Poland (2003) 31–40
11. Rowland, C.H.: Covert channels in the TCP/IP protocol suite. Psionics Technologies (1996) http://www.firstmonday.dk/issues/issue2_5/rowland/.
12. Dunigan, T.: Internet steganography. Technical report, Oak Ridge National Laboratory (Contract No. DE-AC05-96OR22464), Oak Ridge, Tennessee (1998) [ORNL/TM-limited distribution].
13. Rutkowska, J.: The implementation of passive covert channels in the linux kernel. In: 21^{st} Chaos Communication Congress, Berliner Congress Center, Berlin, Germany (2004) www.ccc.de/congress/2004/fahrplan/ files/223-passive-covert-channels-linux.pdf.
14. Abad, C.: Ip checksum covert channels and selected hash collision. http://gray-world.net/cn/papers/ipccc.pdf (2001)
15. Wang, X., Feng, D., Lai, X., Yu, H.: Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMID. http://eprint.iacr.org/2004/199/ (2004)
16. Kaminsky, D.: MD5 to be considered harmful someday. http://www.doxpara.com/md5_someday.pdf (2004)
17. Giffin, J., Greenstadt, R., Litwack, P., Tibbetts, R.: Covert messaging through tcp timestamps. In: Second Workshop on Privacy Enhancing Technologies. Volume 2482 of Lectures Notes in Computer Science., San Francisco, CA, USA, Springer-Verlag Heidelberg (2003) 194–208
18. daemon9 (route@infonexus.com): Loki2 (the implementation). Phrack Magazine, 51, article 6 (1997) http://www.phrack.org/show.php?p=51&a=6.
19. daemon9 (route@infonexus.com), alhambra (alhambra@infornexus.com): Project loki. Phrack Magazine, 49, article 6 (1996) http://www.phrack.org/show.php?p=49&a=6.
20. Skoudis, E.: Counter Hack: A Step-by-Step Guide to Computer Attacks and Effective Defenses. Series in Computer networking and Distributed Systems. Prentice Hall, Upper Saddle River, NJ (2002)
21. Malan, G.R., Watson, D., Jahanian, F., Howell, P.: Transport and application protocol scrubbing. In: Proceedings of the IEEE INFOCOM 2002 Conference, Tel-Aviv, Israel (2000) 1381–1390
22. Handley, M., Paxson, V.: Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics. In: Proceedings of the 10^{th} USENIX Security Symposium, Washington, DC, USA, USENIX Association (2001)
23. Fisk, G., Fisk, M., Papadopoulos, C., Neil, J.: Eliminating steganography in Internet traffic with active wardens. In Oostveen, J., ed.: Information Hiding: Preproceedings of the Fifth International Workshop, Noordwijkerhout, The Netherlands, Springer (2002) 29–46
24. Murdoch, S.J., Lewis, S.: Embedding covert channels into TCP/IP. In: Information Hiding: Proceedings of the Seventh International Workshop, Barcelona, Spain, Springer (2005)
25. Simmons, G.J.: The prisoners' problem and the subliminal channel. In Chaum, D., ed.: Advances in Cryptology, Proceedings of CRYPTO '83, Plenum Press (1984) 51–67
26. Lucena, N.B., Pease, J., Yadollahpour, P., Chapin, S.J.: Syntax and semantics-preserving application-layer protocol steganography. Lecture Notes in Computer Science, Toronto, Canada, Springer-Verlag Heidelberg (2004) 164–179

27. Craver, S.: On public-key steganography in the presence of an active warden. In Aucsmith, D., ed.: Information Hiding: Proceedings of the Second International Workshop. Volume 1525., Portland, OR, USA, Springer (1998) 355–368
28. Katzenbeisser, S., Petitcolas, F.A.: Information Hiding: Techniques for Steganography and Digital Watermarking. Artech House, Norwood, MA (2000)
29. Deering, S., Hinden, R.: Internet protocol, version 6 (ipv6) specification (1998) RFC 2460.
30. Hagen, S.: IPv6 Essentials. First edn. O'Reilly & Associates, Inc., Sebastopol, CA (2002)
31. Kent, S., Atkinson, R.: Security architecture for the internet protocol (1998) RFC 2401.
32. Kent, S., Atkinson, R.: Ip authentication header (1998) RFC 2402.
33. Kent, S., Atkinson, R.: Ip encapsulating security payload (esp) (1998) RFC 2406.
34. Kent, S., Atkinson, R.: Definition of the differentiated services field (ds field) in the ipv4 and ipv6 header (1998) RFC 2402.
35. (IANA), I.A.N.A.: Protocol numbers. http://www.iana.org/assignments/protocol-numbers (2004)
36. (IANA), I.A.N.A.: IP version 6 parameters. http://www.iana.org/assignments/ipv6-parameters (2004)
37. Graf, T.: Messaging over ipv6 destination options. http://net.suug.ch/articles/2003/07/06/ip6msg.html (2003)

A Active Warden Analysis

The channels previously described can be blocked, partially blocked, or remain open depending on the capabilities of the active wardens monitoring the network. The three types of active wardens introduced in section 3, *stateless*, *stateful*, and *network-aware*, apply countermeasures according to their ability.

A.1 Stateless Active Warden

If Wendy is a *stateless* active warden, she knows the protocol syntax and semantics and attempts to verify them. She "sees" one packet at a time. That is, she has no recollection of previous packets nor previously encountered semantic conditions. As a stateless warden, Wendy can perform at two levels of diligence. At the lower one, she checks only that IPv6 headers comply to the specifications. For example, she can ensure that field that are supposed to be zero have exactly that value, otherwise she clears them, enforcing the semantic of the protocol and blocking some possibilities of covert data transmission. This simple stateless active warden can be continuously modifying the traffic without fear of breaking the overt communication. Table 2 shows what Wendy can do to the defined covert channels when she only verifies the protocol semantics.

On the other hand, when Wendy behaves aggressively she also modifies field values that do not necessarily violate the protocol specifications. She can be more effective in blocking covert communication, but may also cause harmful side effects listed in Table 4. The implementation of such active warden requires then careful evaluation of the side effects in the overt communication to avoid a detrimental effect on the network.

A.2 Stateful Active Warden

Wendy acting as a *stateful* active warden can do all the normalizations that a stateless active warden does; she also registers already-observed semantic conditions, and

Table 2. Covert Channels Affected by the Activity of a Stateless Active Warden limited to verifying protocol semantics

Header	**Covert Channel**	**Countermeasure**
Hop-by-Hop	γ	Zeroing all *padding* octets
Hop-by-Hop Routing Authentication	α and β α α	Zeroing values of the *reserved* fields

Table 3. Covert Channels Affected by the Activity of a Stateful Active Warden

Header	**Covert Channel**	**Observed Situation**	**Countermeasure**
IPv6	α	Packets exchanged between two hosts have belong to many different *traffic classes.*	Resetting the *traffic class* to zero in packets exchanged between that pair of hosts.
IPv6	β	Packets traveling between a pair of hosts exhibit a large range of *flow label* values.	Resetting the *flow label* to zero in packets interchanged between the two hosts.
IPv6	ϵ	Variations in the *hop limit* values outside certain range. Wendy can expect fluctuations in the hop limit value because not all packet travel the same path, but not too many.	Resetting the *hop limit* value to be the same (e.g. 255).
Routing	β	Significantly different routing headers attached to packets exchanged between two hosts.	Removing the entire routing header in packets interchanged by the two hosts.
Fragment	γ	Fragment *next header* values differ among fragments belonging to the same packet.	Resetting the *next header* value of all fragments to be the same as the one in the first fragment.
Fragment	δ	Fragments overlap when simulating reassembly.	Dropping all the fragments belonging to the same packet.
Authentication	β	*Sequence number* values do not increase monotonically; also SPI values seem semantically incorrect.	Removing the entire authentication header of the suspicious packet.
ESP	β	*Sequence number* values do not increase monotonically; also SPI values seem semantically incorrect.	Removing the entire ESP header of the suspected packet.

applies that knowledge in subsequent monitoring sessions. Because a stateful active warden can remember previously seen packets, she can more effectively and less invasively eliminate the same covert channels destroyed by a stateless active warden. In

Table 4. Covert Channels Affected by the Activity of a Stateless Active Warden who aggressively attempts to block the covert communication

Header	Covert Channel	Countermeasure	Side Effect
IPv6	α	Resetting the *traffic class* to zero in all packets.	Removes any benefits provided by traffic class aware routers.
IPv6	β	Resetting the *flow label* to zero in all packets.	Removes the router ability of using *flow label* functions. In addition, this action violates the protocol specification, which states that *flow label* values cannot be modified in transit. It is unlikely though that doing it so will cause packet delivery problems because intermediate notes will assume the default behavior.
IPv6	γ	Verifying that the *payload length* value matches the actual datagram payload and removing extra data if found.	Violation of the protocol specification that establishes that extra data must be forwarded, causing the packet to fail during the integrity checking.
IPv6	δ	Verifying that the *next header* value is one of the allowed protocol numbers and stripping the entire header, in case of an unknown value.	Violates the protocol specification and essentially disables the IPv6 header extension mechanism, because any extension header will be unusable until active wardens are aware of new extension header definitions.
IPv6	ϵ	Resetting the *hop limit* value to be the same (e.g. 255)	Risks network congestion. When a routing cycle exists, the active warden will reset the *hop limit* value in every cycle and as a result packets could travel endlessly in the network.
Hop-by-Hop	β	Removing all options of type *routing alert.*	Eliminates the mechanism that informs the router about contents of interest, so it can handle any control data accordingly.
Hop-by-Hop	δ	Discarding all options of *unknown* type.	Violates the protocol specification that says that *unknown* options should be handled accordingly to their *option type* value. By ignoring this requirement, active wardens damage the options header functionality until they learn about new option types.
Routing	β	Removing the entire routing header.	Disregards all advantages of a routing header extension, thus packets cannot travel by predefined paths.

addition, a stateful active warden can detect covert channels that a stateless cannot. Table 3 summarizes what a stateful active warden can do to detect the described covert channels.

A.3 Network-Aware Active Warden

A *network-aware* active warden is the most sophisticated type of warden. Wendy is a stateful warden and also a network topologist. That is, she is able to defeat some of the proposed covert channels because of her knowledge of the topology of the surrounding networks.

If Wendy is a network-aware active warden, she can defeat covert channel β of the routing header (see section 4.3). She can do that taking several countermeasures. She can, for example, verify whether or not the address fields are in fact valid IPv6 addresses. At the same time, she can verify that the addresses are not multicast addresses because they are not allowed to be. In addition, if Wendy has knowledge of the topology of the network where the packet originated, she can match the addresses listed in the routing header to the ones the packet has traversed. The location of a network-aware warden is critical. Being placed in the egress gateway of an organization, Wendy can monitor all outgoing packets and verify the IP addresses in the routing header.