IP Security Protocol Working Group (IPSEC)
INTERNET-DRAFT
Category: Standards track
Expires: December 2002

A. Huttunen
F-Secure Corporation
B. Swander
Microsoft
M. Stenberg
SSH Communications Security Corp
V. Volpe
Cisco Systems
L. DiBurro
Nortel Networks

June 2002

UDP Encapsulation of IPsec Packets
draft-ietf-ipsec-udp-encaps-03.txt

Status of this Memo

Copyright Notice

Abstract

This draft defines methods to encapsulate and decapsulate ESP packets inside UDP packets for the purpose of traversing NATs.

ESP encapsulation as defined in this document is capable of being used in both IPv4 and IPv6 scenarios.

The encapsulation is used whenever negotiated using IKE, as defined in [Kiv02].

Change Log

Version -01

- removed everything related to the AH-protocol
- added instructions on how to use the encapsulation with some other key management protocol than IKE

Version -02

- changed to using 4-byte non-ESP marker, removed all references to using this with other key management protocols
- TCP checksum handling for transport mode related discussion modified
- copied tunnel mode security considerations from the earlier draft-huttunen-ipsec-esp-in-udp-00.txt draft, added transport mode considerations

Version -03

- Clarifications to security considerations

1. Introduction

This draft defines methods to encapsulate and decapsulate ESP packets inside UDP packets for the purpose of traversing NATs. The UDP port numbers are the same as used by IKE traffic, as defined in [Kiv02].

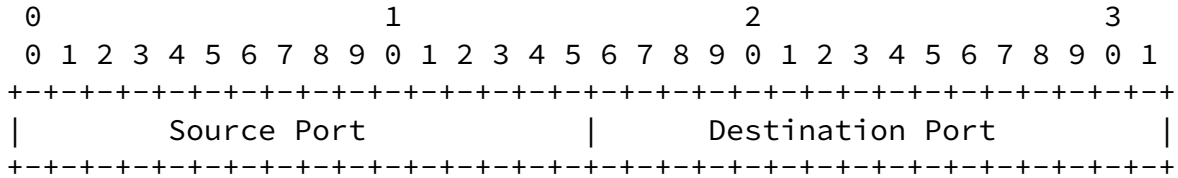
It is up to the need of the clients whether transport mode or tunnel mode is to be supported. L2TP/IPsec clients MUST support transport mode since [RFC 3193] defines that L2TP/IPsec MUST use transport mode], and IPsec tunnel mode clients MUST support tunnel mode.

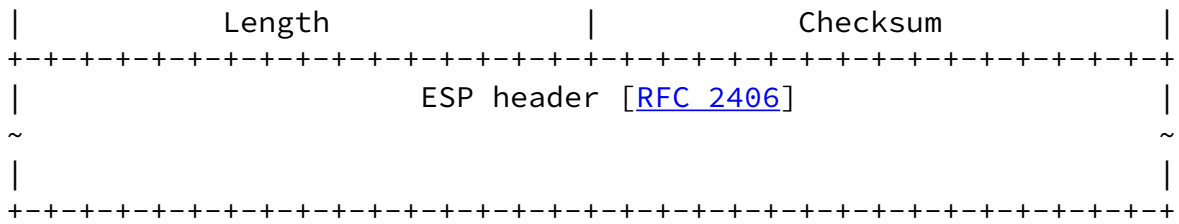
An IKE implementation supporting this draft MUST NOT use the ESP SPI field zero for ESP packets. (XXX To be changed to an IANA allocated SPI value later.) This ensures that IKE packets and ESP packets can be distinguished from each other.

UDP encapsulation of ESP packets as defined in this document is written in terms of IPv4 headers. There is no technical reason why an IPv6 header could not be used as the outer header and/or as the inner header.

2. Packet Formats

2.1 UDP-encapsulated ESP Header Format

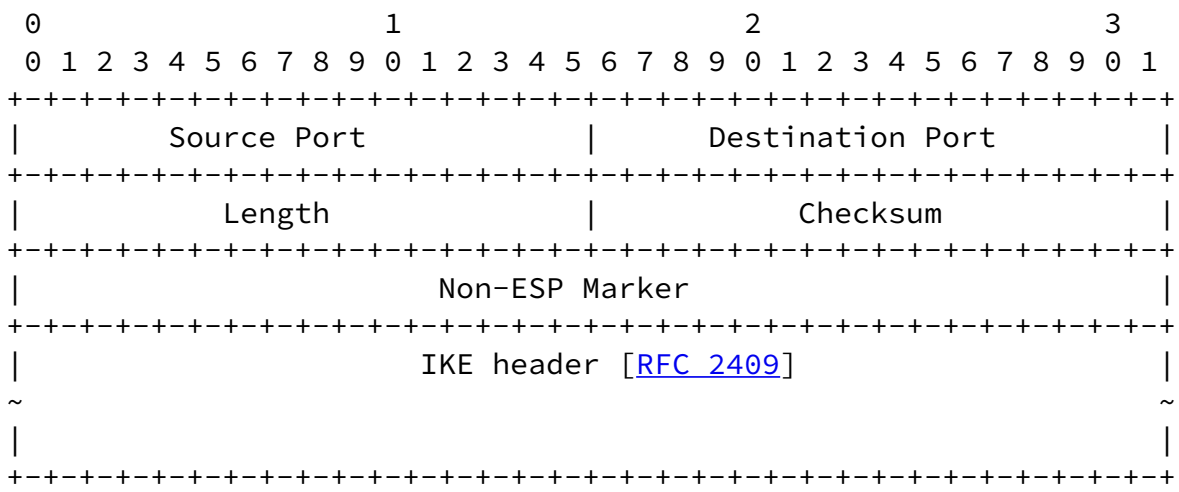




The UDP header is a standard [RFC 768] header, where
- Source Port and Destination Port are the same as used by floated IKE traffic.
- Checksum is zero.

The SPI field in the ESP header must not be zero. (XXX To be changed to an IANA allocated SPI value later.)

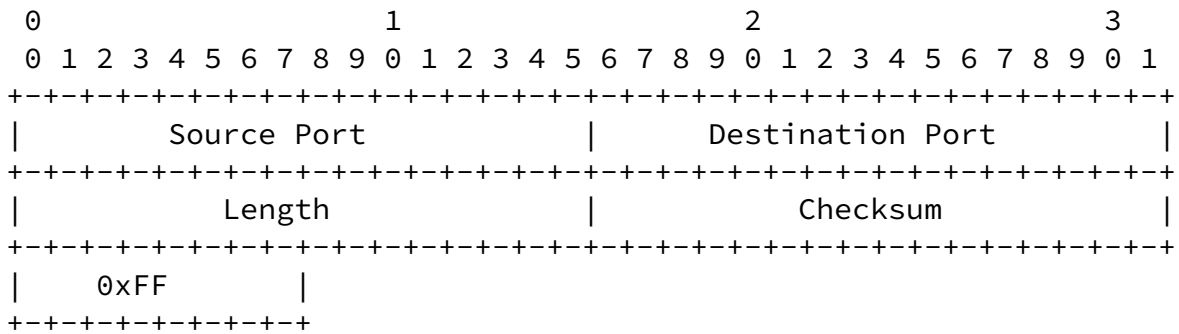
2.2 Floated IKE Header Format



The UDP header is a standard [RFC 768] header, and is used as defined in [Kiv02].

Non-ESP Marker is 4 bytes of zero aligning with the SPI field of an ESP packet. (XXX To be changed to an IANA allocated SPI value later.)

2.3 NAT-keepalive Packet Format



The UDP header is a standard [RFC 768] header, where

- Source Port and Destination Port are the same as used by floated
  IKE traffic.
- Checksum is zero.

The sender SHOULD use a one octet long payload with the value 0xFF.
The receiver SHOULD ignore a received NAT-keepalive packet.

3. Encapsulation and Decapsulation Procedures

3.1 Auxiliary Procedures

3.1.1 Tunnel Mode Decapsulation NAT Procedure

When a tunnel mode has been used to transmit packets, the inner
IP header can contain addresses that are not suitable for the
current network. This procedure defines how these addresses are
to be converted to suitable addresses for the current network.

Depending on local policy, one of the following MUST be done:
a) If a valid source IP address space has been defined in the policy
   for the encapsulated packets from the peer, check that the source
   IP address of the inner packet is valid according to the policy.
b) If an address has been assigned for the remote peer, check
   that the source IP address used in the inner packet is the
   same as the IP address assigned.
c) NAT is performed for the packet, making it suitable for transport
   in the local network.

3.1.2 Transport Mode Decapsulation NAT Procedure

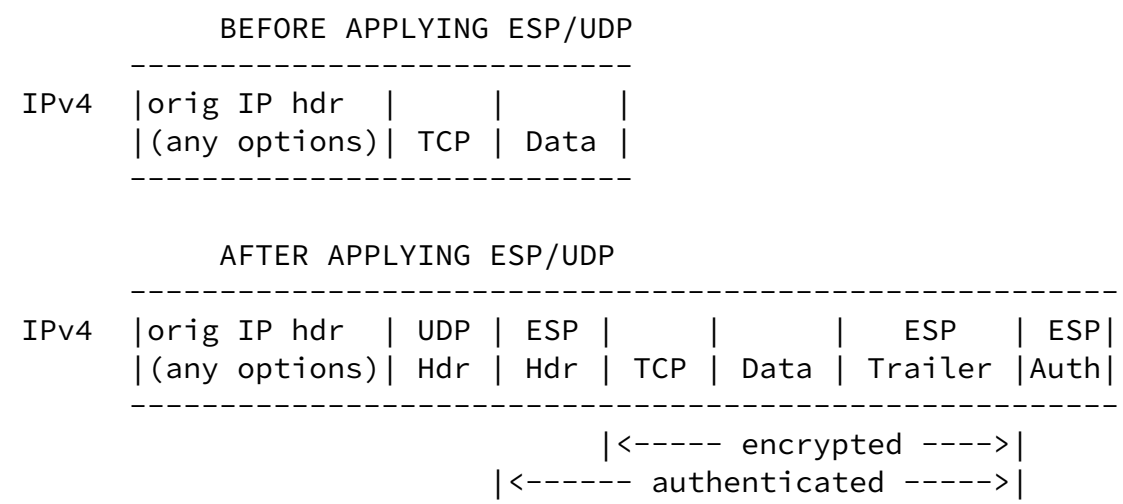
When a transport mode has been used to transmit packets, contained
TCP or UDP headers will contain incorrect checksums due to the change
of parts of the IP header during transit. This procedure defines how
to fix these checksums.

Depending on local policy, one of the following MUST be done:
a) If the protocol header after the ESP header is a TCP/UDP
   header and the peer's real source IP address has been received
   according to [Kiv02], incrementally recompute the TCP/UDP checksum:
   - subtract the IP source address in the received packet
     from the checksum
   - add the real IP source address received via IKE to the checksum
b) If the protocol header after the ESP header is a TCP/UDP
   header, recompute the checksum field in the TCP/UDP header.
c) If the protocol header after the ESP header is an UDP
   header, zero the checksum field in the UDP header. If the protocol
   header after the ESP header is a TCP header, and there is an
   option to flag to the stack that TCP checksum does not need to
   be computed, then that flag MAY be used. This SHOULD only be done
   for transport mode, and if the packet is integrity protected. Tunnel
   mode TCP checksums MUST be verified.

[This is not a violation to the spirit of section 4.2.2.7 in RFC 1122 because a checksum is being generated by the sender, and verified by the receiver. That checksum is the integrity over the packet performed by IPsec.]

In addition an implementation MAY fix any contained protocols that have been broken by NAT.

3.2 Transport Mode ESP Encapsulation

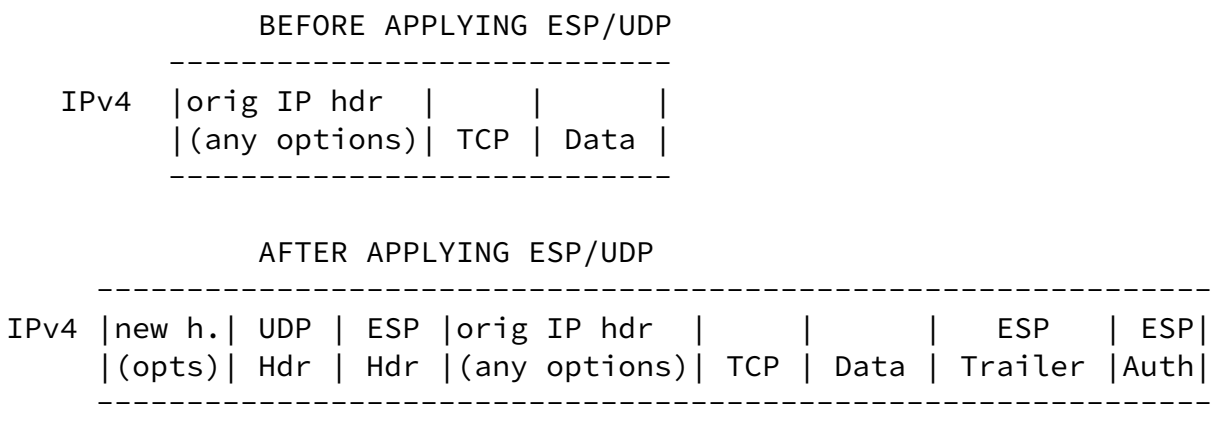


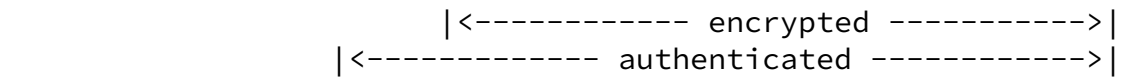
1) Ordinary ESP encapsulation procedure is used.
2) A properly formatted UDP header is inserted where shown.
3) The Total Length, Protocol and Header Checksum fields in the IP header are edited to match the resulting IP packet.

3.3 Transport Mode ESP Decapsulation

1) The UDP header is removed from the packet.
2) The Total Length, Protocol and Header Checksum fields in the new IP header are edited to match the resulting IP packet.
3) Ordinary ESP decapsulation procedure is used.
4) Transport mode decapsulation NAT procedure is used.

3.4 Tunnel Mode ESP Encapsulation





1) Ordinary ESP encapsulation procedure is used.
2) A properly formatted UDP header is inserted where shown.
3) The Total Length, Protocol and Header Checksum fields in the
   new IP header are edited to match the resulting IP packet.

3.5 Tunnel Mode ESP Decapsulation

1) The UDP header is removed from the packet.
2) The Total Length, Protocol and Header Checksum fields in the
   new IP header are edited to match the resulting IP packet.
3) Ordinary ESP decapsulation procedure is used.
4) Tunnel mode decapsulation NAT procedure is used.

4. NAT Keepalive Procedure

The sole purpose of sending NAT-keepalive packets is to keep NAT mappings alive for the duration of a connection between the peers. Reception of NAT-keepalive packets MUST NOT be used to detect liveness of a connection.

A peer MAY send a NAT-keepalive packet if there exists one or more phase I or phase II SAs between the peers, or such an SA has existed at most N minutes earlier. N is a locally configurable parameter with a default value of 5 minutes.

A peer SHOULD send a NAT-keepalive packet if a need to send such packets is detected according to [Kiv02] and if no other packet to the peer has been sent in M seconds. M is a locally configurable parameter with a default value of 20 seconds.

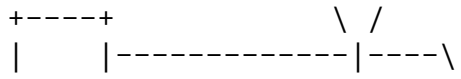
5. Security Considerations

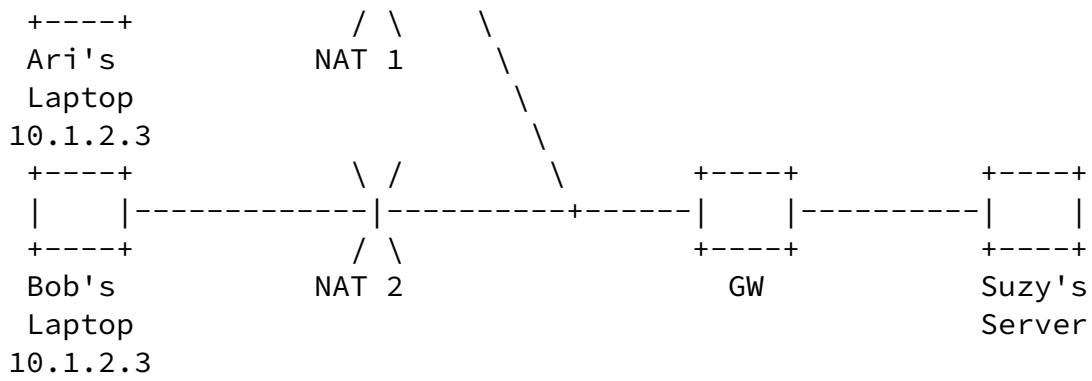
5.1 DoS

On some systems ESPUDP may have DoS attack consequences, especially if ordinary operating system UDP-functionality is being used. It may be recommended not to open an ordinary UDP-port for this.

5.2 Tunnel Mode Conflict

Implementors are warned that it is possible for remote peers to negotiate entries that overlap in a GW, an issue affecting tunnel mode.





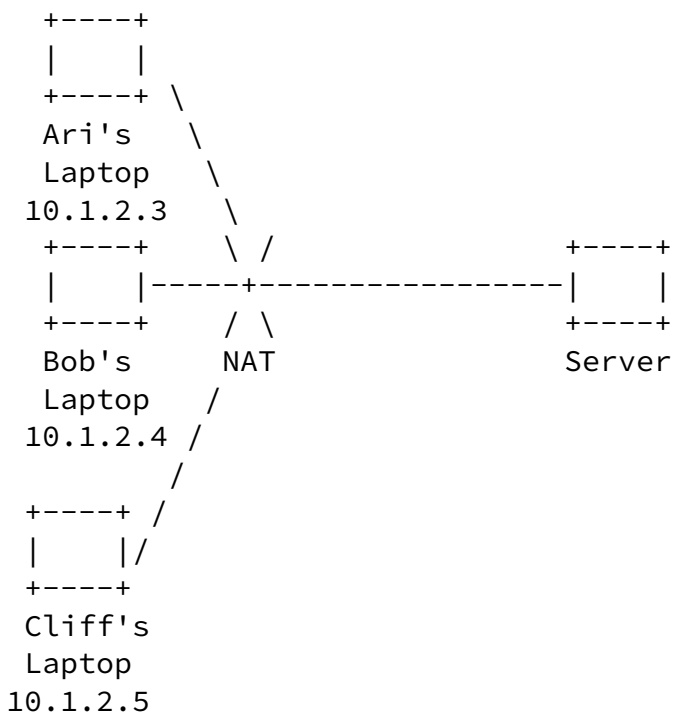
Because GW will now see two possible SAs that lead to 10.1.2.3, it can become confused where to send packets coming from Suzy's server. Implementators MUST devise ways of preventing such a thing from occurring.

It is recommended that GW either assign locally unique IP addresses to A and B using a protocol such as DHCP over IPsec, or uses NAT to change A's and B's source IP addresses to such locally unique addresses before sending packets forward to S.

5.3 Transport Mode Conflict

Another similar issue may occur in transport mode, with 2 clients, Ari and Bob, behind the same NAT talking securely to the same server.

Cliff wants to talk in the clear to the same server.



Now, transport SAs on the server will look like:

To Ari: S to NAT, <traffic desc1>, UDP encap <4500, Y>
To Bob: S to NAT, <traffic desc2>, UDP encap <4500, Z>

Cliff's traffic is in the clear, so there is no SA.

<traffic desc> is the protocol and port information.
The UDP encap ports are the ports used in UDP encapsulated
ESP format of section 2.1. Y,Z are the dynamic ports assigned
by the NAT during the IKE negotiation. So IKE traffic from
Ari's laptop goes out on UDP <4500,4500>. It reaches the server
as UDP <Y,4500>, where Y is the dynamically assigned port.

If the <traffic desc1> overlaps <traffic desc2>, then
simple filter lookups may not be sufficient to determine
which SA needs to be used to send traffic. Implementations
MUST handle this situation, either by disallowing
conflicting connections, or by other means.

Assume now that Cliff wants to connect to the server S in the
clear. This is going to be difficult to configure since
the server already has a policy from S to the NAT's external
address, for securing <traffic desc>. For totally non-overlapping
traffic descriptions, this is possible.

Sample server policy could be:
To Ari: S to NAT, All UDP, secure
To Bob: S to NAT, All TCP, secure
To Cliff: S to NAT, ALL ICMP, clear text

Note, this policy also lets Ari and Bob send cleartext ICMP to the
server.

The server sees all clients behind the NAT as the same IP address,
so setting up different policies for the same traffic descriptor
is in principle impossible.

A problematic example configuration on the server is:

S to NAT, TCP, secure (for Ari and Bob)
S to NAT, TCP, clear (for Cliff)

The problem is that the server cannot enforce his policy, since it
is possible that misbehaving Bob sends traffic in the clear. This
is indistinguishable from Cliff sending traffic in the clear.
So it is impossible to guarantee security from some clients behind
a NAT, and also allow clear text from different clients behind the
SAME NAT. If the server's security policy allows, however, it can
do best effort security: if the client from behind the NAT
initiates security, his connection will be secured. If he sends
in the clear, the server will still accept that clear text.

So, for security guarantees, the above problematic scenario MUST NOT be allowed on servers. For best effort security, this scenario MAY be used.

6. Intellectual Property Rights

The IETF has been notified of intellectual property rights claimed in regard to some or all of the specification contained in this document. For more information consult the online list of claimed rights.

SSH Communications Security Corp has notified the working group of one or more patents or patent applications that may be relevant to this internet-draft. SSH Communications Security Corp has already given a licence for those patents to the IETF. For more information consult the online list of claimed rights.

7. Acknowledgments

Thanks to Tero Kivinen and William Dixon who contributed actively to this document.

Thanks to Joern Sierwald, Tamir Zegman, Tatu Ylonen and Santeri Paavolainen who contributed to the previous drafts about NAT traversal.

8. References

[RFC 768] Postel, J., "User Datagram Protocol", August 1980

[RFC 1122] R. Braden (Editor), "Requirements for Internet Hosts -- Communication Layers", October 1989

[RFC-2119] Bradner, S., "Key words for use in RFCs to indicate Requirement Levels", March 1997

[RFC 2406] Kent, S., "IP Encapsulating Security Payload (ESP)", November 1998

[RFC 2409] D. Harkins, D. Carrel, "The Internet Key Exchange (IKE)", November 1998

[RFC 3193] Patel, B. et. al, "Securing L2TP using IPsec", November 2001

[Kiv02] Kivinen, T. et. al., draft-ietf-ipsec-nat-t-ike-02.txt, "Negotiation of NAT-Traversal in the IKE", April 2002

9. Authors' Addresses

Ari Huttunen

F-Secure Corporation
Tammasaarenkatu 7
FIN-00181 HELSINKI
Finland
E-mail: Ari.Huttunen@F-Secure.com

Brian Swander
Microsoft
One Microsoft Way
Redmond WA 98052
E-mail: briansw@microsoft.com

Markus Stenberg
SSH Communications Security Corp
Fredrikinkatu 42
FIN-00100 HELSINKI
Finland
E-mail: mstenber@ssh.com

Victor Volpe
Cisco Systems
124 Grove Street
Suite 205
Franklin, MA 02038
E-mail: vvolpe@cisco.com

Larry DiBurro
Nortel Networks
80 Central Street
Boxborough, MA 01719
ldiburro@nortelnetworks.com