Network
Internet-Draft
Intended status: Standards Track
Expires: January 8, 2017

T. Pauly
Apple Inc.
S. Touati
Ericsson
R. Mantha
Cisco Systems
July 7, 2016

TCP Encapsulation of IKE and IPSec Packets
draft-ietf-ipsecme-tcp-encaps-01

Abstract

This document describes a method to transport IKE and IPSec packets over a TCP connection for traversing network middleboxes that may block IKE negotiation over UDP. This method, referred to as TCP encapsulation, involves sending all packets for tunnel establishment as well as tunneled packets over a TCP connection. This method is intended to be used as a fallback option when IKE cannot be negotiated over UDP.

Table of Contents

1. Introduction

IKEv2 [RFC7296] is a protocol for establishing IPSec tunnels, using IKE messages over UDP for control traffic, and using Encapsulating Security Payload (ESP) messages for tunneled data traffic. Many network middleboxes that filter traffic on public hotspots block all

UDP traffic, including IKE and IPSec, but allow TCP connections through since they appear to be web traffic. Devices on these networks that need to use IPSec (to access private enterprise networks, to route voice-over-IP calls to carrier networks, or because of security policies) are unable to establish IPSec tunnels. This document defines a method for encapsulating both the IKE control messages as well as the IPSec data messages within a TCP connection.

Using TCP as a transport for IPSec packets adds a third option to the list of traditional IPSec transports:

1. Direct. Currently, IKE negotiations begin over UDP port 500. If no NAT is detected between the initiator and the receiver, then subsequent IKE packets are sent over UDP port 500 and IPSec data packets are sent using ESP [RFC4303].

2. UDP Encapsulation [RFC3948]. If a NAT is detected between the initiator and the receiver, then subsequent IKE packets are sent over UDP port 4500 with four bytes of zero at the start of the UDP payload and ESP packets are sent out over UDP port 4500. Some peers default to using UDP encapsulation even when no NAT are detected on the path as some middleboxes do not support IP protocols other than TCP and UDP.

3. TCP Encapsulation. If both of the other two methods are not available or appropriate, both IKE negotiation packets as well as ESP packets can be sent over a single TCP connection to the peer.

Direct use of ESP or UDP Encapsulation should be preferred by IKE implementations due to performance concerns when using TCP Encapsulation Section 12. Most implementations should use TCP Encapsulation only on networks where negotiation over UDP has been attempted without receiving responses from the peer, or if a network is known to not support UDP.

1.1. Prior Work and Motivation

Encapsulating IKE connections within TCP streams is a common approach to solve the problem of UDP packets being blocked by network middleboxes. The goal of this document is to promote interoperability by providing a standard method of framing IKE and ESP message within streams, and to provide guidelines for how to configure and use TCP encapsulation.

Some previous solutions include:

Cellular Network Access Interworking Wireless LAN (IWLAN) uses IKEv2 to create secure connections to cellular carrier networks for making voice calls and accessing other network services over Wi-Fi networks. 3GPP has recommended that IKEv2 and ESP packets be sent within a TLS connection to be able to establish connections on restrictive networks.

ISAKMP over TCP Various non-standard extensions to ISAKMP have been deployed that send IPSec traffic over TCP or TCP-like packets.

SSL VPNs Many proprietary VPN solutions use a combination of TLS and IPSec in order to provide reliability.

IKEv2 over TCP IKEv2 over TCP as described in [I-D.nir-ipsecme-ike-tcp] is used to avoid UDP fragmentation.

1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Configuration

One of the main reasons to use TCP encapsulation is that UDP traffic may be entirely blocked on a network. Because of this, support for TCP encapsulation is not specifically negotiated in the IKE exchange. Instead, support for TCP encapsulation must be pre-configured on both the initiator and the responder.

The configuration defined on each peer should include the following parameters:

- o One or more TCP ports on which the responder will listen for incoming connections. Note that the initiator may initiate TCP connections to the responder from any local port.

- o Optionally, an extra framing protocol to use on top of TCP to further encapsulate the stream of IKE and IPSec packets. See Appendix A for a detailed discussion.

This document leaves the selection of TCP ports up to implementations. It is suggested to use TCP port 4500, which is allocated for IPSec NAT Traversal.

Since TCP encapsulation of IKE and IPSec packets adds overhead and has potential performance trade-offs compared to direct or UDP-encapsulated tunnels (as described in Performance Considerations,

Section 12), implementations SHOULD prefer ESP direct or UDP encapsulated tunnels over TCP encapsulated tunnels when possible.

3. TCP-Encapsulated Header Formats

In order to encapsulate IKE and ESP messages within a TCP stream, a 16-bit length field precedes every message. If the first 32-bits of the message are zeros (a Non-ESP Marker), then the contents comprise an IKE message. Otherwise, the contents comprise an ESP message. Authentication Header (AH) messages are not supported for TCP encapsulation.

Although a TCP stream may be able to send very long messages, implementations SHOULD limit message lengths to typical UDP datagram ESP payload lengths. The maximum message length is used as the effective MTU for connections that are being encrypted using ESP, so the maximum message length will influence characteristics of inner connections, such as the TCP Maximum Segment Size (MSS).

3.1. TCP-Encapsulated IKE Header Format

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Non-ESP Marker                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                      IKE header [RFC7296]                     ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1

The IKE header is preceded by a 16-bit length field in network byte order that specifies the length of the IKE packet within the TCP stream. As with IKE over UDP port 4500, a zeroed 32-bit Non-ESP Marker is inserted before the start of the IKE header in order to differentiate the traffic from ESP traffic between the same addresses and ports.

- o Length (2 octets, unsigned integer) - Length of the IKE packet including the Length Field and Non-ESP Marker.

3.2. TCP-Encapsulated ESP Header Format

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                      ESP header [RFC4303]                     ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2

The ESP header is preceded by a 16-bit length field in network byte order that specifies the length of the ESP packet within the TCP stream.

The SPI field in the ESP header MUST NOT be a zero value.

- o Length (2 octets, unsigned integer) - Length of the ESP packet including the Length Field.

4. TCP-Encapsulated Stream Prefix

Each stream of bytes used for IKE and IPSec encapsulation MUST begin with a fixed sequence of six bytes as a magic value, containing the characters "IKETCP" as ASCII values. This allows peers to differentiate this protocol from other protocols that may be run over TCP streams, since the bytes do not overlap with the valid start of any other known stream protocol. This value is only sent once, by the Initiator only, at the beginning of any stream of IKE and ESP messages.

If other framing protocols are used within TCP to further encapsulate or encrypt the stream of IKE and ESP messages, the Stream Prefix must be at the start of the Initiator's IKE and ESP message stream within the added protocol layer [Appendix A].

```
 0      1      2      3      4      5
+------+------+------+------+------+------+
| 0x49 | 0x4b | 0x45 | 0x54 | 0x43 | 0x50 |
+------+------+------+------+------+------+
```

Figure 3

5. Applicability

TCP encapsulation is applicable only when it has been configured to be used with specific IKE peers. If a responder is configured to use TCP encapsulation, it MUST listen on the configured port(s) in case any peers will initiate new IKE sessions. Initiators MAY use TCP encapsulation for any IKE session to a peer that is configured to support TCP encapsulation, although it is recommended that initiators should only use TCP encapsulation when traffic over UDP is blocked.

Since the support of TCP encapsulation is a configured property, not a negotiated one, it is recommended that if there are multiple IKE endpoints representing a single peer (such as multiple machines with different IP addresses when connecting by Fully-Qualified Domain Name, or endpoints used with IKE redirection), all of the endpoints equally support TCP encapsulation.

If TCP encapsulation is being used for a specific IKE SA, all messages for that IKE SA and its Child SAs MUST be sent over a TCP connection until the SA is deleted or MOBIKE is used to change the SA endpoints and/or encapsulation protocol. No packets should be sent over UDP or direct ESP for the IKE SA or its Child SAs while using TCP encapsulation.

6. Connection Establishment and Teardown

When the IKE initiator uses TCP encapsulation for its negotiation, it will initiate a TCP connection to the responder using the configured TCP port. The first bytes sent on the stream MUST be the stream prefix value [Section 4]. After this prefix, encapsulated IKE messages will negotiate the IKE SA and initial Child SA [RFC7296]. After this point, both encapsulated IKE Figure 1 and ESP Figure 2 messages will be sent over the TCP connection.

In order to close an IKE session, either the initiator or responder SHOULD gracefully tear down IKE SAs with DELETE payloads. Once all SAs have been deleted, the initiator of the original connection MUST close the TCP connection.

An unexpected FIN or a RST on the TCP connection may indicate either a loss of connectivity, an attack, or some other error. If a DELETE payload has not been sent, both sides SHOULD maintain the state for their SAs for the standard lifetime or time-out period. The original initiator (that is, the endpoint that initiated the TCP connection and sent the first IKE_SA_INIT message) is responsible for re-establishing the TCP connection if it is torn down for any unexpected reason. Since new TCP connections may use different ports due to NAT

mappings or local port allocations changing, the responder MUST allow packets for existing SAs to be received from new source ports.

A peer MUST discard a partially received message due to a broken connection.

The streams of data sent over any TCP connection used for this protocol MUST begin with the stream prefix value followed by a complete message, which is either an encapsulated IKE or ESP message. If the connection is being used to resume a previous IKE session, the responder can recognize the session using either the IKE SPI from an encapsulated IKE message or the ESP SPI from an encapsulated ESP message. If the session had been fully established previously, it is suggested that the initiator send an UPDATE_SA_ADDRESSES message if MOBIKE is supported, or an INFORMATIONAL message (a keepalive) otherwise. If either initiator or responder receives a stream that cannot be parsed correctly, it MUST close the TCP connection.

Multiple TCP connections between the initiator and the responder are allowed, but their use must take into account the initiator capabilities and the deployment model such as to connect to multiple gateways handling different ESP SAs when deployed in a high availability model. It is also possible to negotiate multiple IKE SAs over the same TCP connection.

The processing of the TCP packets is the same whether its within a single or multiple TCP connections.

7. Interaction with NAT Detection Payloads

When negotiating over UDP port 500, IKE_SA_INIT packets include NAT_DETECTION_SOURCE_IP and NAT_DETECTION_DESTINATION_IP payloads to determine if UDP encapsulation of IPSec packets should be used. These payloads contain SHA-1 digests of the SPIs, IP addresses, and ports. IKE_SA_INIT packets sent on a TCP connection SHOULD include these payloads, and SHOULD use the applicable TCP ports when creating and checking the SHA-1 digests.

If a NAT is detected due to the SHA-1 digests not matching the expected values, no change should be made for encapsulation of subsequent IKE or ESP packets, since TCP encapsulation inherently supports NAT traversal. Implementations MAY use the information that a NAT is present to influence keep-alive timer values.

8. Using MOBIKE with TCP encapsulation

When an IKE session is transitioned between networks using MOBIKE [RFC4555], the initiator of the transition may switch between using TCP encapsulation, UDP encapsulation, or no encapsulation. Implementations that implement both MOBIKE and TCP encapsulation MUST support dynamically enabling and disabling TCP encapsulation as interfaces change.

The encapsulation method of ESP packets MUST always match the encapsulation method of the IKE negotiation, which may be different when an IKE endpoint changes networks. When a MOBIKE-enabled initiator changes networks, the UPDATE_SA_ADDRESSES notification SHOULD be sent out first over UDP before attempting over TCP. If there is a response to the UPDATE_SA_ADDRESSES notification sent over UDP, then the ESP packets should be sent directly over IP or over UDP port 4500 (depending on if a NAT was detected), regardless of if a connection on a previous network was using TCP encapsulation. Similarly, if the responder only responds to the UPDATE_SA_ADDRESSES notification over TCP, then the ESP packets should be sent over the TCP connection, regardless of if a connection on a previous network did not use TCP encapsulation.

9. Using IKE Message Fragmentation with TCP encapsulation

IKE Message Fragmentation [RFC7383] is not required when using TCP encapsulation, since a TCP stream already handles the fragmentation of its contents across packets. Since fragmentation is redundant in this case, implementations might choose to not negotiate IKE fragmentation. Even if fragmentation is negotiated, an implementation MAY choose to not fragment when going over a TCP connection.

If an implementation supports both MOBIKE and IKE fragmentation, it SHOULD negotiate IKE fragmentation over a TCP encapsulated session in case the session switches to UDP encapsulation on another network.

10. Considerations for Keep-alives and DPD

Encapsulating IKE and IPSec inside of a TCP connection can impact the strategy that implementations use to detect peer liveness and to maintain middlebox port mappings. Peer liveness should be checked using IKE Informational packets [RFC7296].

In general, TCP port mappings are maintained by NATs longers than UDP port mappings, so IPSec ESP NAT keep-alives [RFC3948] SHOULD NOT be sent when using TCP encapsulation. Any implementation using TCP encapsulation MUST silently drop incoming NAT keep-alive packets, and

not treat them as errors. NAT keep-alive packets over a TCP encapsulated IPSec connection will be sent with a length value of 1 byte, whose value is 0xFF [Figure 2].

Note that depending on the configuration of TCP and TLS on the connection, TCP keep-alives [RFC1122] and TLS keep-alives [RFC6520] may be used. These MUST NOT be used as indications of IKE peer liveness.

11. Middlebox Considerations

Many security networking devices such as Firewalls or Intrusion Prevention Systems, network optimization/acceleration devices and Network Address Translation (NAT) devices keep the state of sessions that traverse through them.

These devices commonly track the transport layer and/or the application layer data to drop traffic that is anomalous or malicious in nature.

A network device that monitors up to the application layer will commonly expect to see HTTP traffic within a TCP socket running over port 80, if non-HTTP traffic is seen (such as TCP encapsulated IKE), this could be dropped by the security device.

A network device that monitors the transport layer will track the state of TCP sessions, such as TCP sequence numbers. TCP encapsulation of IKE should therefore use standard TCP behaviors to avoid being dropped by middleboxes.

12. Performance Considerations

Several aspects of TCP encapsulation for IKE and IPSec packets may negatively impact the performance of connections within the tunnel. Implementations should be aware of these and take these into consideration when determining when to use TCP encapsulation.

12.1. TCP-in-TCP

If the outer connection between IKE peers is over TCP, inner TCP connections may suffer effects from using TCP within TCP. In particular, the inner TCP's round-trip-time estimation will be affected by the burstiness of the outer TCP. This will make loss-recovery of the inner TCP traffic less reactive and more prone to spurious retransmission timeouts.

12.2. Added Reliability for Unreliable Protocols

Since ESP is an unreliable protocol, transmitting ESP packets over a TCP connection will change the fundamental behavior of the packets. Some application-level protocols that prefer packet loss to delay (such as Voice over IP or other real-time protocols) may be negatively impacted if their packets are retransmitted by the TCP connection due to packet loss.

12.3. Quality of Service Markings

Quality of Service (QoS) markings, such as DSCP and Traffic Class, should be used with care on TCP connections used for encapsulation. Individual packets SHOULD NOT use different markings than the rest of the connection, since packets with different priorities may be routed differently and cause unnecessary delays in the connection.

12.4. Maximum Segment Size

A TCP connection used for IKE encapsulation SHOULD negotiate its maximum segment size (MSS) in order to avoid unnecessary fragmentation of packets.

13. Security Considerations

IKE responders that support TCP encapsulation may become vulnerable to new Denial-of-Service (DoS) attacks that are specific to TCP, such as SYN-flooding attacks. Responders should be aware of this additional attack-surface.

Attackers may be able to disrupt the TCP connection by sending spurious RST packets. Due to this, implementations SHOULD make sure that IKE session state persists even if the underlying TCP connection is torn down.

14. IANA Considerations

This memo includes no request to IANA.

TCP port 4500 is already allocated to IPSec. This port MAY be used for the protocol described in this document, but implementations MAY prefer to use other ports based on local policy.

15. Acknowledgments

The authors would like to acknowledge the input and advice of Stuart Cheshire, Delziel Fernandes, Yoav Nir, Christoph Paasch, Yaron Sheffer, David Schinazi, Graham Bartlett, Byju Pularikkal, March Wu

and Kingwel Xie. Special thanks to Eric Kinnear for his
implementation work.

16. References

16.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <http://www.rfc-editor.org/info/rfc7296>.

16.2. Informative References

[I-D.nir-ipsecme-ike-tcp]
Nir, Y., "A TCP transport for the Internet Key Exchange", draft-nir-ipsecme-ike-tcp-01 (work in progress), July 2012.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <http://www.rfc-editor.org/info/rfc1122>.

[RFC2817] Khare, R. and S. Lawrence, "Upgrading to TLS Within HTTP/1.1", RFC 2817, DOI 10.17487/RFC2817, May 2000, <http://www.rfc-editor.org/info/rfc2817>.

[RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, DOI 10.17487/RFC3948, January 2005, <http://www.rfc-editor.org/info/rfc3948>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <http://www.rfc-editor.org/info/rfc4303>.

[RFC4555] Eronen, P., "IKEv2 Mobility and Multihoming Protocol (MOBIKE)", RFC 4555, DOI 10.17487/RFC4555, June 2006, <http://www.rfc-editor.org/info/rfc4555>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC6520] Seggelmann, R., Tuexen, M., and M. Williams, "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension", RFC 6520, DOI 10.17487/RFC6520, February 2012, <http://www.rfc-editor.org/info/rfc6520>.

[RFC7383] Smyslov, V., "Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation", RFC 7383, DOI 10.17487/RFC7383, November 2014, <http://www.rfc-editor.org/info/rfc7383>.

Appendix A. **Using TCP encapsulation with TLS**

This section provides recommendations on the support of TLS with the TCP encapsulation.

When using TCP encapsulation, implementations may choose to use TLS [RFC5246], to be able to traverse middle-boxes, which may block non HTTP traffic.

If a web proxy is applied to the ports for the TCP connection, and TLS is being used, the initiator can send an HTTP CONNECT message to establish a tunnel through the proxy [RFC2817].

The use of TLS should be configurable on the peers. The responder may expect to read encapsulated IKE and ESP packets directly from the TCP connection, or it may expect to read them from a stream of TLS data packets. The initiator should be pre-configured to use TLS or not when communicating with a given port on the responder.

When new TCP connections are re-established due to a broken connection, TLS must be re-negotiated. TLS Session Resumption is recommended to improve efficiency in this case.

The security of the IKE session is entirely derived from the IKE negotiation and key establishment and not from the TLS session (which in this context is only used for encapsulation purposes), therefore when TLS is used on the TCP connection, both the initiator and responder SHOULD allow the NULL cipher to be selected for performance reasons.

Implementations should be aware that the use of TLS introduces another layer of overhead requiring more bytes to transmit a given

IKE and IPSec packet. For this reason, direct ESP, UDP encapsulation, or TCP encapsulation without TLS should be preferred in situations in which TLS is not required in order to traverse middle-boxes.

Appendix B. Example exchanges of TCP Encapsulation with TLS

B.1. Establishing an IKE session

```
                Client                               Server
              ----------                           ----------
     1)  --------------------  TCP Connection  -------------------
         (IP_I:Port_I  -> IP_R:TCP443 or TCP4500)
         TcpSyn                    ---------->
                                   <----------         TcpSyn,Ack
         TcpAck                    ---------->

     2)  ---------------------  TLS Session  ---------------------
         ClientHello               ---------->
                                                      ServerHello
                                                     Certificate*
                                               ServerKeyExchange*
                                   <----------    ServerHelloDone
         ClientKeyExchange
         CertificateVerify*
         [ChangeCipherSpec]
         Finished                  ---------->
                                               [ChangeCipherSpec]
                                   <----------           Finished

     3)  ---------------------- Stream Prefix --------------------
         "IKETCP"                  ---------->
     4)  ----------------------- IKE Session ---------------------
         IKE_SA_INIT               ---------->
         HDR, SAi1, KEi, Ni,
         [N(NAT_DETECTION_*_IP)]
                                   <----------        IKE_SA_INIT
                                              HDR, SAr1, KEr, Nr,
                                           [N(NAT_DETECTION_*_IP)]
         first IKE_AUTH            ---------->
         HDR, SK {IDi, [CERTREQ]
         CP(CFG_REQUEST), IDr,
         SAi2, TSi, TSr, ...}
                                   <----------     first IKE_AUTH
                                       HDR, SK {IDr, [CERT], AUTH,
                                              EAP, SAr2, TSi, TSr}
         EAP                       ---------->
         repeat 1..N times
                                   <----------                EAP
         final IKE_AUTH            ---------->
         HDR, SK {AUTH}
                                   <----------     final IKE_AUTH
                                     HDR, SK {AUTH, CP(CFG_REPLY),
                                               SA, TSi, TSr, ...}
         ----------------- IKE Tunnel Established ----------------
```

Figure 4

1. Client establishes a TCP connection with the server on port 443 or 4500.

2. Client initiates TLS handshake. During TLS handshake, the server SHOULD NOT request the client's' certificate, since authentication is handled as part of IKE negotiation.

3. Client send the Stream Prefix for TCP encapsulated IKE [Section 4] traffic to signal the beginning of IKE negotation.

4. Client and server establish an IKE connection. This example shows EAP-based authentication, although any authentication type may be used.

B.2. Deleting an IKE session

```
              Client                                Server
            ----------                            ----------
  1)  ----------------------- IKE Session ---------------------
      INFORMATIONAL             ---------->
      HDR, SK {[N,] [D,]
             [CP,] ...}
                                <----------       INFORMATIONAL
                                             HDR, SK {[N,] [D,]
                                                      [CP], ...}

  2)  ---------------------  TLS Session  ---------------------
      close_notify              ---------->
                                <----------        close_notify
  3)  --------------------  TCP Connection  -------------------
      TcpFin                    ---------->
                                <----------                 Ack
                                <----------              TcpFin
      Ack                       ---------->
      --------------------- Tunnel Deleted  -------------------
```

Figure 5

1. Client and server exchange INFORMATIONAL messages to notify IKE SA deletion.

2. Client and server negotiate TLS session deletion using TLS CLOSE_NOTIFY.

3. The TCP connection is torn down.

Unless the TCP connection and/or TLS session are being used for multiple IKE SAs, the deletion of the IKE SA should lead to the disposal of the underlying TLS and TCP state.

B.3. Re-establishing an IKE session

```
                Client                                Server
              ----------                            ----------
  1) --------------------  TCP Connection  -------------------
     (IP_I:Port_I  -> IP_R:TCP443 or TCP4500)
     TcpSyn                    ---------->
                               <----------          TcpSyn,Ack
     TcpAck                    ---------->
  2) ---------------------  TLS Session  ---------------------
     ClientHello               ---------->
                               <----------         ServerHello
                                           [ChangeCipherSpec]
                                                      Finished
     [ChangeCipherSpec]        ---------->
     Finished
  3) ---------------------- Stream Prefix --------------------
     "IKETCP"                  ---------->
  4) <---------------------> IKE/ESP flow <------------------>
```

Figure 6

1. If a previous TCP connection was broken (for example, due to a RST), the client is responsible for re-initiating the TCP connection. The initiator's address and port (IP_I and Port_I) may be different from the previous connection's address and port.

2. In ClientHello TLS message, the client SHOULD send the Session ID it received in the previous TLS handshake if available. It is up to the server to perform either an abbreviated handshake or full handshake based on the session ID match.

3. After TCP and TLS are complete, the client sends the Stream Prefix for TCP encapsulated IKE traffic [Section 4].

4. The IKE and ESP packet flow can resume. If MOBIKE is being used, the initiator SHOULD send UPDATE_SA_ADDRESSES.

B.4. Using MOBIKE between UDP and TCP Encapsulation

```
                 Client                                    Server
               ----------                                ----------
         (IP_I1:UDP500 -> IP_R:UDP500)
     1)  ----------------- IKE_SA_INIT Exchange -----------------
         (IP_I1:UDP4500 -> IP_R:UDP4500)
         Intial IKE_AUTH          ----------->
         HDR, SK { IDi, CERT, AUTH,
         CP(CFG_REQUEST),
         SAi2, TSi, TSr,
         N(MOBIKE_SUPPORTED) }
                                  <-----------    Initial IKE_AUTH
                                        HDR, SK { IDr, CERT, AUTH,
                                              EAP, SAr2, TSi, TSr,
                                             N(MOBIKE_SUPPORTED) }
         <---------------- IKE tunnel establishment ------------->

     2)  ------------ MOBIKE Attempt on new network --------------
         (IP_I2:UDP4500 -> IP_R:UDP4500)
         INFORMATIONAL            ----------->
         HDR, SK { N(UPDATE_SA_ADDRESSES),
         N(NAT_DETECTION_SOURCE_IP),
         N(NAT_DETECTION_DESTINATION_IP) }


     3)  --------------------  TCP Connection  -------------------
         (IP_I2:PORT_I -> IP_R:TCP443 or TCP4500)
         TcpSyn                   ----------->
                                  <-----------          TcpSyn,Ack
         TcpAck                   ----------->

     4)  ---------------------  TLS Session  ---------------------
         ClientHello              ----------->
                                                       ServerHello
                                                      Certificate*
                                                ServerKeyExchange*
                                  <-----------     ServerHelloDone
         ClientKeyExchange
         CertificateVerify*
         [ChangeCipherSpec]
         Finished                 ----------->
                                                [ChangeCipherSpec]
                                  <-----------            Finished
     5)  ---------------------- Stream Prefix --------------------
         "IKETCP"                  ---------->
```

```
   6)  ----------------------- IKE Session ---------------------
       INFORMATIONAL            ----------->
       HDR, SK { N(UPDATE_SA_ADDRESSES),
       N(NAT_DETECTION_SOURCE_IP),
       N(NAT_DETECTION_DESTINATION_IP) }

                                &lt;-----------       INFORMATIONAL
                           HDR, SK { N(NAT_DETECTION_SOURCE_IP),
                               N(NAT_DETECTION_DESTINATION_IP) }
   7)  &lt;----------------- IKE/ESP data flow ------------------->
```

Figure 7

1. During the IKE_SA_INIT exchange, the client and server exchange MOBIKE_SUPPORTED notify payloads to indicate support for MOBIKE.

2. The client changes its point of attachment to the network, and receives a new IP address. The client attempts to re-establish the IKE session using the UPDATE_SA_ADDRESSES notify payload, but the server does not respond because the network blocks UDP traffic.

3. The client brings up a TCP connection to the server in order to use TCP encapsulation.

4. The client initiates and TLS handshake with the server.

5. The client sends the Stream Prefix for TCP encapsulated IKE traffic [Section 4].

6. The client sends the UPDATE_SA_ADDRESSES notify payload on the TCP encapsulated connection.

7. The IKE and ESP packet flow can resume.

Authors' Addresses

Tommy Pauly
Apple Inc.
1 Infinite Loop
Cupertino, California 95014
US

Email: tpauly@apple.com

Samy Touati
Ericsson
300 Holger Way
San Jose, California 95134
US

Email: samy.touati@ericsson.com

Ravi Mantha
Cisco Systems
SEZ, Embassy Tech Village
Panathur, Bangalore 560 037
India

Email: ramantha@cisco.com