Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 7, 2013

Y. Nir
Check Point
December 4, 2012

A TCP transport for the Internet Key Exchange
draft-ietf-ipsecme-ike-tcp-01

Abstract

This document describes using TCP for IKE messages. This facilitates the transport of large messages over paths where fragments are either dropped, or where packet loss makes the use of large UDP packets unreliable.

Status of this Memo

Copyright Notice

1. Introduction

The Internet Key Exchange version 2 (IKEv2), specified in [RFC5996] uses UDP to transport the exchange messages. Some of those messages may be fairly large. Specifically, the messages of the IKE_AUTH exchange can become quite large, as they may contain a chain of certificates, an "Auth" payload (that may contain a public key signature), CRLs, and some configuration information that is carried in the CFG payload.

When such UDP packets exceed the path MTU, they get fragmented. This increases the probability of packets being dropped. The retransmission mechanisms in IKE (as described in section 2.1 of RFC 5996) takes care of that as long as packet loss is at a reasonable level. More recently we have seen a number of service providers dropping fragmented packets. Firewalls and NAT devices need to keep state for each packet where some (but not all) of the fragments have passed through. This creates a burden in terms of memory, especially for high capacity devices such as Carrier-Grade NAT (CGN) or high capacity firewalls.

The BEHAVE working group has an Internet Draft describing required behavior of CGNs ([I-D.ietf-behave-lsn-requirements]). It requires CGNs to comply with [RFC4787], which in section 11 requires NAT devices to support fragments. However, some people deploying IKE have found that some ISPs have begun to drop fragments in preparation for deploying CGNs. While we all hope for a future where all devices comply with the emerging standards, or even a future where CGNs are not required, we have to make IKE work today.

The solution described in this document is to transport the IKE messages over a TCP ([RFC0793]) connection rather than over UDP. IKE packets describe their own length, so they are well-suited for transport over a stream-based connection such as TCP. The Initiator opens a TCP connection to the Responder's port 500, sends the requests and receives the responses, and then closes the connection. TCP can handle arbitrary-length messages, works well with any sized data, and is well supported by all ISP infrastructure.

1.1. Non-Goals of this Specification

Firewall traversal is not a goal of this specification. If a firewall has a policy to block IKE and/or IPsec, hiding the IKE exchange in TCP is not expected to help. Some implementations hide both IKE and IPsec in a TCP connection, usually pretending to be HTTPS by using port 443. This has a significant impact on bandwidth and gateway capacity, and even this is defeated by better firewalls. SSL VPNs tunnel IP packets over TLS, but the latest firewalls are

also TLS proxies, and are able to defeat this as well.

This document is not part of that arms race. It is only meant to allow IKE to work When faced with broken infrastructure that drops large IP packets.

1.2. Conventions Used in This Document

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 [RFC2119].

2. The Protocol

2.1. Initiator

An Initiator MAY try IKE using TCP for any request. It opens a TCP connection from an arbitrary port to port 500 of the Responder. When the three-way handshake completes, the Initiator MUST send the request. If the Initiator knows that this request is the last request needed at this time, it MAY half-close the TCP connection, or it MAY wait until the last response has been received. When all responses have been received, the Initiator MUST close the connection. If the peer has closed the connection before all requests have been transmitted or responded to, the Initiator SHOULD either open a new TCP connection or transmit them over UDP again.

An initiator MUST accept responses sent over IKE within the same connection, but MUST also accept responses over other transports, if the request had been sent over them as well.

An initiator that is configured to respond to IKE over TCP on some port, and is not prevented from receiving TCP connections by network address translation (see Section 3.2), MUST send an IKE_TCP_SUPPORTED notification (Section 2.5) in the Initial request.

Note that stateless cookies may be dependent on some of the parameters of the connection, so retransmitting the IKE_INITIAL request with a stateless cookie over a different transport may cause the cookie to be invalid. For this reason, retransmissions with a cookie SHOULD be sent over the same transport.

2.2. Responder

A Responder MAY accept TCP connections to port 500, and if it does, it MUST accept IKE requests over this connection. Responses to requests received over this connection MUST also go over this

connection. If the connection has closed before the Responder has had a chance to respond, it MUST NOT respond over UDP, but MUST instead wait for a retransmission over UDP or over another TCP connection.

The responder MUST accept different requests on different transports. Specifically, the Responder MUST NOT rely on subsequent requests coming over the same transport. For example, it is entirely acceptable to have the IKE_INITIAL exchange come over UDP port 500, while the IKE_AUTH request comes over TCP, and some following requests might come over UDP port 4500 (because NAT has been detected).

A responder that is configured to support IKE over TCP and receives an IKEv2 Initial request over any other transport MUST send an IKE_TCP_SUPPORTED notification (Section 2.5) in the Initial response. the responder MAY send this notification even if the Initial request was received over TCP.

If the responder has some requests of its own to send, it MUST NOT use a connection that has been opened by a peer. Instead, it MUST either use UDP or else open a new TCP connection to the original Initiator's TCP port, specified in the IKE_TCP_SUPPORTED notification in the Initial request. If the Initial request did not include this notification, the original Responder MUST NOT initiate IKE over TCP to the original Initiator.

The normal flow of things is that the Initiator opens a connection and closes its side first. The responder closes after sending the last response where the initiator has already half-closed the connection. If, however, a significant amount of time has passed, and neither new requests arrive nor the connection is closed by the initiator, the Responder MAY close or even reset the connection.

This specification makes no recommendation as to how long such a timeout should be, but a few seconds should be enough.

The stateless cookie mechanism in IKEv2 only assures that the initiator is able to respond to the address and port of the request. TCP already provides this with the three-way handshake. If the IKE_INITIAL exchange is transmitted over TCP, the stateless cookie mechanism SHOULD NOT be used.

2.3. Transmitter

The transmitter, whether an initiator transmitting a request or a responder transmitting a response MUST NOT retransmit over the same connection. TCP takes care of that. It SHOULD send the IKE header

and the IKE payloads with a single command or in rapid succession, because the receiver might block on reading from the socket.

2.4. Receiver

The IKE header is copied from RFC 5996 below for reference:

```
                        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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       IKE SA Initiator's SPI                  |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       IKE SA Responder's SPI                  |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Next Payload | MjVer | MnVer | Exchange Type |     Flags     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          Message ID                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                            Length                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: IKE Header Format

The receiver MUST first read in the 28 bytes that make up the IKE header. The Responder then subtracts 28 from the length field, and reads the resulting number of bytes. The combined message, comprised on 28 header bytes and whatever number of payload bytes is processed the same way as regular UDP messages. That includes retransmission detection, with one slight difference: if a retransmitted request is detected, the response is retransmitted as well, but using the current TCP connection rather than whatever other transport had been used for the original transmission of the request.

2.5. IKE_TCP_SUPPORTED Notification

This notification is sent by a responder over non-TCP transports to inform the initiator that this specification is supported and configured.

The Notify payload is formatted as follows:

```
                        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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ! Next Payload  !C!  RESERVED   !         Payload Length        !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !  Protocol ID  !   SPI Size    !IKE_TCP_SUPPORTED Message Type !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           TCP Port            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- Protocol ID (1 octet) MUST be 0.
- SPI Size (1 octet) MUST be zero, in conformance with section 3.10 of RFC 5996.
- IKE_TCP_SUPPORTED Notify Message Type (2 octets) - MUST be xxxxx, the value assigned for IKE_TCP_SUPPORTED. TBA by IANA.
- TCP port (2 octets) - The TCP port to which the recipient should open TCP connections. This is not necessarily the same port that the IKE gateway is listening to. See Section 3.2. If the sender is not subject to network address translation, the port SHOULD be 500.

3. Operational Considerations

Most IKE messages are relatively short. All but the IKE_AUTH exchange in IKEv2 are comprised of short messages that fit in a single packet on most networks. The Informational exchange could be an exception, as it may contain arbitrary-length CFG payloads, but in practice this is not done. It is only the IKE_AUTH exchange that has long messages. UDP has advantages in lower latency and lower resource consumption, so it makes sense to use UDP whenever TCP is not required.

The requirements in Section 2.2 were written so that different requests may be sent over different transports. The initiator can choose the transport on a per-request basis. So one obvious policy would be to do everything over UDP except the specific requests that tend to become too big. This way the first messages use UDP, and the Initiator can set up the TCP connection at the same time, eliminating the latency penalty of using TCP. This may not always be the most efficient policy, though. It means that the first messages sent over TCP are relatively large ones, and TCP slow start may cause an extra roundtrip, because the message has exceeded the transmission window. An initiator using this policy MUST NOT go to TCP if the responder has not indicated support by sending the IKE_TCP_SUPPORTED notification (Section 2.5) in the Initial response.

An alternative method, that is probably easier for the Initiator to

implement, is to do an entire "mission" using the same transport. So if TCP is needed for long messages and an IKE SA has not yet been created, the Initiator will open a TCP connection, and perform all 2-4 requests needed to set up a child SA over the same connection.

Yet another policy would be to begin by using UDP, and at the same time set up the TCP connection. If at any point the TCP handshake completes, the next requests go over that connection. This method can be used to auto-discover support of TCP on the responder. This is easier for the user than configuring which peers support TCP, but has the potential of wasting resources, as TCP connections may finish the three-way handshake just when IKE over UDP has finished. The requirements from the responder ensure that all these policies will work.

3.1. Liveness Check

The TCP connections described in this document are short-lived. We do not expect them to stay for the lifetime of the SA, but to get torn down by either side within seconds of the SA being set up. Because of this, they are not well-suited for the transport of short requests such as those for liveness check.

Although liveness checks MAY be sent over TCP, this is not recommended.

On the other hand, see Section 3.2 for when liveness check should be used.

3.2. Network Address Translation

If the IKE gateway is subject to network address translation (NAT), TCP ports may be translated, so that one port on the NAT device gets translated to some other port on the gateway. In this case, the gateway MUST advertise the NAT device port in the IKE_TCP_SUPPORTED notification.

In some cases, the NAT or some other box prevents incoming TCP connections to the IKE peer behind it. In these cases, the IKE peer MUST NOT advertise support using the IKE_TCP_SUPPORTED notification.

When IKE peers detect the presence of a NAT device during the IKE exchange, they typically switch to working over UDP port 4500. Sending the IKE_AUTH messages over this UDP port creates a port mapping entry on the NAT device, and this mapping can then be used for bidirectional traffic between the peers. When using IKE over TCP, this mapping is not created, so traffic can only flow from the initiator to the responder. To make a bidirectional mapping, it is

RECOMMENDED that when NAT is detected, initiators initiate a liveness check using UDP 4500 to the responders immediately following the successful IKE_AUTH exchange.

4. Security Considerations

Most of the security considerations for IKE over TCP are the same as those for UDP as in RFC 5996.

For the Responder, listening to TCP port 500 involves all the risks of maintaining any TCP server. Precautions against DoS attacks, such as SYN cookies are RECOMMENDED. see [RFC4987] for details.

5. IANA Considerations

IANA is requested to assign a notify message type from the status types range (16418-40959) of the "IKEv2 Notify Message Types" registry with name "IKE_TCP_SUPPORTED"

No IANA action is required for the TCP port, as TCP port 500 is already allocated to "ISAKMP".

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.

6.2. Informative References

[I-D.ietf-behave-lsn-requirements] Perreault, S., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common requirements for Carrier Grade NATs (CGNs)", draft-ietf-behave-lsn-requirements-09 (work in progress), August 2012.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC4787] Audet, F. and C. Jennings, "Network Address Translation

(NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.

[RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, August 2007.

Author's Address

Yoav Nir
Check Point Software Technologies Ltd.
5 Hasolelim st.
Tel Aviv 67897
Israel

Email: ynir@checkpoint.com