ADD
Internet-Draft
Intended status: Standards Track
Expires: March 13, 2021

M. Boucadair
Orange
T. Reddy
McAfee
D. Wing
Citrix
V. Smyslov
ELVIS-PLUS

September 9, 2020

Internet Key Exchange Protocol Version 2 (IKEv2) Configuration for Encrypted DNS
draft-btw-add-ipsecme-ike-01

Abstract

This document specifies a new Internet Key Exchange Protocol Version 2 (IKEv2) Configuration Payload Attribute Types for encrypted DNS with a focus on DNS-over-HTTPS (DoH), DNS-over-TLS (DoT), and DNS-over-QUIC (DoQ).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 13, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

This document specifies encrypted DNS configuration for an Internet Key Exchange Protocol Version 2 (IKEv2) [RFC7296] initiator, particularly the Authentication Domain Name (ADN, defined in [RFC8310]) of DNS-over-HTTPS (DoH) [RFC8484], DNS-over-TLS (DoT) [RFC7858], or DNS-over-QUIC (DoQ) [I-D.ietf-dprive-dnsoquic].

This document introduces new IKEv2 Configuration Payload Attribute Types (Section 4) for the support of DoT, DoH, and DoQ DNS servers.

This document targets the deployments discussed in Section 3.3 of [I-D.box-add-requirements]. Sample use cases are discussed in Section 3. The Configuration Payload Attribute Types defined in this document are not specific to these deployments, but can also be used in other deployment contexts.

Note that, for many years, typical designs have often considered that the DNS server was usually located inside the protected domain, but could be located outside of it. With DoH, DoT, or DoQ the latter option becomes plausible.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

This document makes use of the terms defined in [RFC8499] and [I-D.ietf-dnsop-terminology-ter].

Also, this document makes use of the terms defined in [RFC7296]. In particular, readers should be familiar with "initiator" and "responder" terms used in that document.

Do53 refers to unencrypted DNS.

Encrypted DNS refers to as scheme where DNS messages are sent over an encrypted channel. Examples of encrypted DNS are DoT, DoH, and DoQ.

ENCDNS_IP*_* refers to any IKEv2 Configuration Payload Attribute Types defined in Section 4.

ENCDNS_IP4_* refers to an IKEv2 Configuration Payload Attribute Type that carries one or multiple IPv4 addresses of an encrypted DNS server.

ENCDNS_IP6_* refers to an IKEv2 Configuration Payload Attribute Type that carries one or multiple IPv6 addresses of an encrypted DNS server.

3. Sample Deployment Scenarios

3.1. Roaming Enterprise Users

In this Enterprise scenario (Section 1.1.3 of [RFC7296]), a roaming user connects to the Enterprise network through an IPsec tunnel. The split-tunnel Virtual Private Network (VPN) configuration allows the endpoint to access hosts that resides in the Enterprise network [RFC8598] using that tunnel; other traffic not destined to the Enterprise does not traverse the tunnel. In contrast, a non-split-tunnel VPN configuration causes all traffic to traverse the tunnel into the enterprise.

For both split- and non-split-tunnel configurations, the use of encrypted DNS instead of Do53 provides privacy and integrity protection along the entire path (rather than just to the VPN

termination device) and can communicate the encrypted DNS server policies.

For split-tunnel VPN configurations, the endpoint uses the Enterprise-provided encrypted DNS server to resolve internal-only domain names.

For non-split-tunnel VPN configurations, the endpoint uses the Enterprise-provided encrypted DNS server to resolve both internal and external domain names.

Enterprise networks are susceptible to internal and external attacks. To minimize that risk all enterprise traffic is encrypted (Section 2.1 of [I-D.arkko-farrell-arch-model-t]).

3.2. VPN Service Provider

Legacy VPN service providers usually preserve end-users' data confidentiality by sending all communication traffic through an encrypted tunnel. A VPN service provider can also provide guarantees about the security of the VPN network by filtering malware and phishing domains.

Browsers and OSes support DoH/DoT; VPN providers may no longer expect DNS clients to fallback to Do53 just because it is a closed network.

The encrypted DNS server hosted by the VPN service provider can be securely discovered by the endpoint using the IKEv2 Configuration Payload Attribute Type.

3.3. DNS Offload

VPN service providers typically allow split-tunnel VPN configuration in which users can choose applications that can be excluded from the tunnel. For example, users may exclude applications that restrict VPN access.

The encrypted DNS server hosted by the VPN service provider can be securely discovered by the endpoint using the IKEv2 Configuration Payload Attribute Type.

4. IKEv2 Configuration Payload Attribute Types for Encrypted DNS

The ENCDNS_IP*_* IKEv2 Configuration Payload Attribute Types are used to configure a DoT, DoH, or DoQ DNS server. All these attributes share the format shown in Figure 1.

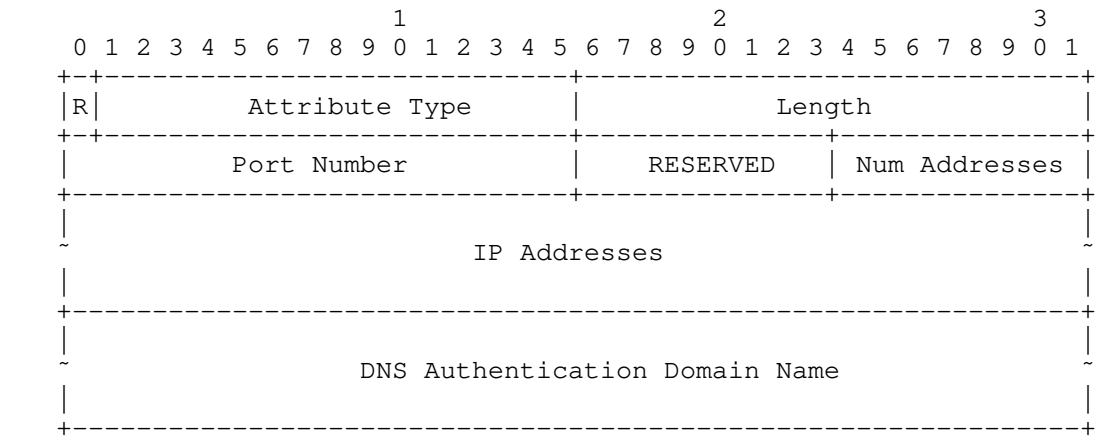


Figure 1: Attributes Format

The fields of the attribute shown in Figure 1 are as follows:

- R (Reserved, 1 bit) - This bit MUST be set to zero and MUST be ignored on receipt (see Section 3.15.1 of [RFC7296] for details).

- Attribute Type (15 bits) - Identifier for Configuration Attribute Type; is set to one of the values listed in Section 8.1.

- Length (2 octets, unsigned integer) - Length of the data in octets. In particular, this field is set to:

  * 0 if the Configuration payload has types CFG_REQUEST or CFG_ACK.

  * (2 + Length of the ADN + N * 4) for ENCDNS_IP4_* attributes if the Configuration payload of a has types CFG_REPLY or CFG_SET; N being the number of included IPv4 addresses ('Num addresses').

  * (2 + Length of the ADN + N * 16) for ENCDNS_IP6_* attributes if the Configuration payload has types CFG_REPLY or CFG_SET; N being the number of included IPv6 addresses ('Num addresses').

- Port Number (2 octets, unsigned integer) - Indicates the port number to be used for the encrypted DNS. As a reminder, the default port number is 853 for DoT and 443 for DoH.

- RESERVED (1 octet) - These bits are reserved for future use. These bits MUST be set to zero by the sender and MUST be ignored by the receiver.

- Num Addresses (1 octet) - Indicates the number of enclosed IPv4 (for ENCDNS_IP4_* attribute types) or IPv6 (for ENCDNS_IP6_* attribute types) addresses.

- IP Address(es) (variable) - One or more IPv4 or IPv6 addresses to be used to reach the encrypted DNS identified by the name in the DNS Authentication Domain Name.

- Authentication Domain Name (variable) - A fully qualified domain name of the DoT, DoH, or DoQ DNS server following the syntax defined in [RFC5890]. The name MUST NOT contain any terminators (e.g., NULL, CR).

  An example of valid ADN for DoH server is "doh1.example.com".

5. IKEv2 Protocol Exchange

This section describes how an initiator can be configured with an encrypted DNS server (e.g., DoH, DoT) using IKEv2.

Initiators indicate the support of an encrypted DNS in the CFG_REQUEST payloads by including one or multiple ENCDNS_IP*_* attributes, while responders supply the encrypted DNS configuration in the CFG_REPLY payloads. Concretely:

If the initiator supports encrypted DNS, it includes one or more ENCDNS_IP*_* attributes in the CFG_REQUEST with the "Attribute Type" set to the requested encrypted DNS type (Section 4). For each supported encrypted DNS type the initiator MUST include exactly one attribute with the Length field set to 0, so that no data is included for these attributes.

For each ENCDNS_IP*_* attribute from the CFG_REQUEST, if the responder supports the corresponding encrypted DNS type, and absent any policy, the responder sends back an ENCDNS_IP*_* attribute in the CFG_REPLY with this encrypted DNS type and an appropriate list of IP addresses, a port number, and an ADN. The list of IP addresses MUST NOT be empty. Multiple instances of the same ENCDNS_IP*_* attribute MAY be returned if distinct ADNs (or port numbers) are to be returned by the responder.

If the CFG_REQUEST includes an ENCDNS_IP*_* attribute but the CFG_REPLY does not include an ENCDNS_IP*_* matching the requested encrypted DNS type, this is an indication that requested encrypted DNS type(s) is not supported by the responder or the responder is not configured to provide corresponding server addresses.

The behavior of the responder if it receives both ENCDNS_IP*_* and INTERNAL_IP6_DNS (or INTERNAL_IP4_DNS) attributes is policy-based and deployment-specific. However, it is RECOMMENDED that if the responder includes at least one ENCDNS_IP*_* attribute in the reply, it should not include any of INTERNAL_IP4_DNS/ INTERNAL_IP6_DNS attributes.

The DNS client establishes an encrypted DNS session (e.g., DoT, DoH, DoQ) with the address(es) conveyed in ENCDNS_IP*_* and uses the mechanism discussed in Section 8 of [RFC8310] to authenticate the DNS server certificate using the authentication domain name conveyed in ENCDNS_IP*_*.

If the IPsec connection is a split-tunnel configuration and the initiator negotiated INTERNAL_DNS_DOMAIN as per [RFC8598], the DNS client MUST resolve the internal names using ENCDNS_IP*_* DNS servers.

Note: [RFC8598] requires INTERNAL_IP6_DNS (or INTERNAL_IP4_DNS) attribute to be mandatory present when INTERNAL_DNS_DOMAIN is included. This specification relaxes that constraint in the presence of ENCDNS_IP*_* attributes.

6. URI Template

DoH servers may support more than one URI Template [RFC8484]. Also, if the resolver hosts several DoH services (e.g., no-filtering, blocking adult content, blocking malware), these services can be discovered as templates.

Upon discovery of a DoH resolver (Section 5), the DoH client contacts that DoH resolver to retrieve the list of supported DoH services using the well-known URI defined in [I-D.btw-add-rfc8484-clarification]. DoH clients re-iterates that request regularly to retrieve an updated list of supported DoH services.

How a DoH client makes use of the configured DoH services is out of the scope of this document.

7. Security Considerations

This document adheres to the security considerations defined in [RFC7296]. In particular, this document does not alter the trust on the DNS configuration provided by a responder.

Networks are susceptible to internal attacks as discussed in Section 3.2 of [I-D.arkko-farrell-arch-model-t]. Hosting encrypted

DNS server even in case of split-VPN configuration minimizes the attack vector (e.g., a compromised network device cannot monitor/ modify DNS traffic). This specification describes a mechanism to restrict access to the DNS messages to only the parties that need to know.

In most deployment scenarios, the initiator expects that it is using the encrypted DNS server hosted by a specific organization or enterprise. The DNS client can validate the signatory (i.e., cryptographically attested by the organization hosting the encrypted DNS server) using, for example, [I-D.reddy-add-server-policy-selection], and the user can review human-readable privacy policy information of the DNS server and assess whether the DNS server performs DNS-based content filtering. This helps to protect from a compromised IKE server advertising a malicious encrypted DNS server.

The initiator may trust the encrypted DNS servers supplied by means of IKEv2 from a trusted responder more than the locally provided DNS servers, especially in the case of connecting to unknown or untrusted networks (e.g., coffee shops or hotel networks).

If the encrypted DNS server that was discovered by means of IKEv2 does not meet the privacy preserving data policy and filtering requirements of the user, the user can instruct the DNS client to take appropriate actions. For example, the action can be to use the local encrypted DNS server only to access internal-only DNS names and use another DNS server (that addresses his/her expectations) for public domains. Such actions and their handling is out of scope.

If IKEv2 responder has used NULL Authentication method [RFC7619] to authenticate itself, the initiator MUST NOT use returned ENCDNS_IP*_* servers configuration unless it is pre-configured in the OS or the browser.

This specification does not extend the scope of accepting DNSSEC trust anchors beyond the usage guidelines defined in Section 6 of [RFC8598].

8. IANA Considerations

8.1. Configuration Payload Attribute Types

This document requests IANA to assign the following new IKEv2 Configuration Payload Attribute Types from the "IKEv2 Configuration Payload Attribute Types" namespace available at https://www.iana.org/assignments/ikev2-parameters/ ikev2-parameters.xhtml#ikev2-parameters-21.

Value	Attribute Type	Multi-Valued	Length	Reference
TBA1	ENCDNS_IP4_DOT	YES	0 or more	RFC XXXX
TBA2	ENCDNS_IP6_DOT	YES	0 or more	RFC XXXX
TBA3	ENCDNS_IP4_DOH	YES	0 or more	RFC XXXX
TBA4	ENCDNS_IP6_DOH	YES	0 or more	RFC XXXX
TBA5	ENCDNS_IP4_DOQ	YES	0 or more	RFC XXXX
TBA6	ENCDNS_IP6_DOQ	YES	0 or more	RFC XXXX

9. Acknowledgements

Many thanks to Yoav Nir, Christian Jacquenet, Paul Wouters, and Tommy Pauly for the review and comments.

Yoav and Paul suggested the use of one single attribute carrying both the name and an IP address instead of depending on the existing INTERNAL_IP6_DNS and INTERNAL_IP4_DNS attributes.

Christian Huitema suiggested to return a port number in the attributes.

10. References

10.1. 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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/info/rfc5890>.

[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, <https://www.rfc-editor.org/info/rfc7296>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8310] Dickinson, S., Gillmor, D., and T. Reddy, "Usage Profiles for DNS over TLS and DNS over DTLS", RFC 8310, DOI 10.17487/RFC8310, March 2018, <https://www.rfc-editor.org/info/rfc8310>.

10.2. Informative References

[I-D.arkko-farrell-arch-model-t]
Arkko, J. and S. Farrell, "Challenges and Changes in the Internet Threat Model", draft-arkko-farrell-arch-model-t-04 (work in progress), July 2020.

[I-D.box-add-requirements]
Box, C., Pauly, T., Wood, C., and T. Reddy.K, "Requirements for Adaptive DNS Discovery", draft-box-add-requirements-00 (work in progress), September 2020.

[I-D.btw-add-rfc8484-clarification]
Boucadair, M., Cook, N., Reddy.K, T., and D. Wing, "Supporting Redirection for DNS Queries over HTTPS (DoH)", draft-btw-add-rfc8484-clarification-02 (work in progress), July 2020.

[I-D.ietf-dnsop-terminology-ter]
Hoffman, P., "Terminology for DNS Transports and Location", draft-ietf-dnsop-terminology-ter-02 (work in progress), August 2020.

[I-D.ietf-dprive-dnsoquic]
Huitema, C., Mankin, A., and S. Dickinson, "Specification of DNS over Dedicated QUIC Connections", draft-ietf-dprive-dnsoquic-00 (work in progress), April 2020.

[I-D.reddy-add-server-policy-selection]
Reddy.K, T., Wing, D., Richardson, M., and M. Boucadair, "DNS Server Selection: DNS Server Information with Assertion Token", draft-reddy-add-server-policy-selection-04 (work in progress), July 2020.

[RFC7619] Smyslov, V. and P. Wouters, "The NULL Authentication Method in the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 7619, DOI 10.17487/RFC7619, August 2015, <https://www.rfc-editor.org/info/rfc7619>.

[RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <https://www.rfc-editor.org/info/rfc7858>.

[RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <https://www.rfc-editor.org/info/rfc8484>.

[RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <https://www.rfc-editor.org/info/rfc8499>.

[RFC8598] Pauly, T. and P. Wouters, "Split DNS Configuration for the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 8598, DOI 10.17487/RFC8598, May 2019, <https://www.rfc-editor.org/info/rfc8598>.

Authors' Addresses

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Tirumaleswar Reddy
McAfee, Inc.
Embassy Golf Link Business Park
Bangalore, Karnataka 560071
India

Email: TirumaleswarReddy_Konda@McAfee.com

Dan Wing
Citrix Systems, Inc.
USA

Email: dwing-ietf@fuggles.com

Valery Smyslov
ELVIS-PLUS
RU

Email: svan@elvis.ru

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 19, 2021

C. Hopps
LabN Consulting, L.L.C.

November 15, 2020

IP Traffic Flow Security
draft-ietf-ipsecme-iptfs-03

Abstract

This document describes a mechanism to enhance IPsec traffic flow security by adding traffic flow confidentiality to encrypted IP encapsulated traffic. Traffic flow confidentiality is provided by obscuring the size and frequency of IP traffic using a fixed-sized, constant-send-rate IPsec tunnel. The solution allows for congestion control as well as non-constant send-rate usage.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 19, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

Traffic Analysis ([RFC4301], [AppCrypt]) is the act of extracting information about data being sent through a network. While one may directly obscure the data through the use of encryption [RFC4303], the traffic pattern itself exposes information due to variations in it's shape and timing ([I-D.iab-wire-image], [AppCrypt]). Hiding the size and frequency of traffic is referred to as Traffic Flow Confidentiality (TFC) per [RFC4303].

[RFC4303] provides for TFC by allowing padding to be added to encrypted IP packets and allowing for transmission of all-pad packets (indicated using protocol 59). This method has the major limitation that it can significantly under-utilize the available bandwidth.

The IP-TFS solution provides for full TFC without the aforementioned bandwidth limitation. This is accomplished by using a constant-send-rate IPsec [RFC4303] tunnel with fixed-sized encapsulating packets; however, these fixed-sized packets can contain partial, whole or multiple IP packets to maximize the bandwidth of the tunnel. A non-constant send-rate is allowed, but the confidentiality properties of its use are outside the scope of this document.

For a comparison of the overhead of IP-TFS with the RFC4303 prescribed TFC solution see Appendix C.

Additionally, IP-TFS provides for dealing with network congestion [RFC2914]. This is important for when the IP-TFS user is not in full control of the domain through which the IP-TFS tunnel path flows.

1.1. Terminology & Concepts

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document assumes familiarity with IP security concepts described in [RFC4301].

2. The IP-TFS Tunnel

As mentioned in Section 1 IP-TFS utilizes an IPsec [RFC4303] tunnel (SA) as it's transport. To provide for full TFC, fixed-sized encapsulating packets are sent at a constant rate on the tunnel.

The primary input to the tunnel algorithm is the requested bandwidth of the tunnel. Two values are then required to provide for this bandwidth, the fixed size of the encapsulating packets, and rate at which to send them.

The fixed packet size may either be specified manually or can be determined through the use of Path MTU discovery [RFC1191] and [RFC8201].

Given the encapsulating packet size and the requested tunnel bandwidth, the corresponding packet send rate can be calculated. The packet send rate is the requested bandwidth divided by the payload size of the encapsulating packet.

The egress of the IP-TFS tunnel MUST allow for and expect the ingress (sending) side of the IP-TFS tunnel to vary the size and rate of sent encapsulating packets, unless constrained by other policy.

2.1. Tunnel Content

As previously mentioned, one issue with the TFC padding solution in [RFC4303] is the large amount of wasted bandwidth as only one IP packet can be sent per encapsulating packet. In order to maximize bandwidth IP-TFS breaks this one-to-one association.

IP-TFS aggregates as well as fragments the inner IP traffic flow into fixed-sized encapsulating IPsec tunnel packets. Padding is only added to the the tunnel packets if there is no data available to be sent at the time of tunnel packet transmission, or if fragmentation has been disabled by the receiver.

This is accomplished using a new Encapsulating Security Payload (ESP, [RFC4303]) type which is identified by the number IPTFS_PROTOCOL (Section 6.1).

2.2. IPTFS_PROTOCOL Payload Content

The IPTFS_PROTOCOL payload content defined in this document is comprised of a 4 or 24 octet header followed by either a partial, a full or multiple partial or full data blocks. The following diagram illustrates this IPTFS_PROTOCOL payload within the ESP packet. See Section 6.1 for the exact formats of the IPTFS_PROTOCOL payload.

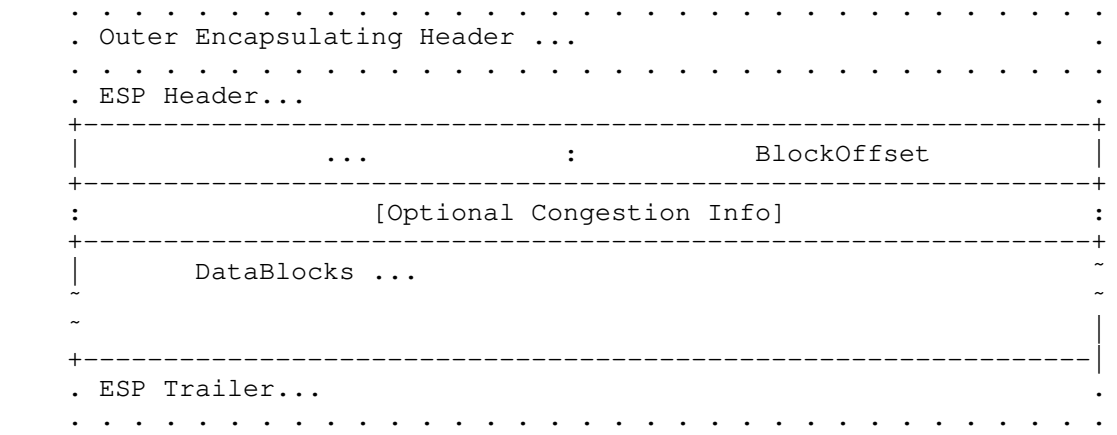


Figure 1: Layout of an IP-TFS IPsec Packet

The "BlockOffset" value is either zero or some offset into or past the end of the "DataBlocks" data.

If the "BlockOffset" value is zero it means that the "DataBlocks" data begins with a new data block.

Conversely, if the "BlockOffset" value is non-zero it points to the start of the new data block, and the initial "DataBlocks" data belongs to a previous data block that is still being re-assembled.

The "BlockOffset" can point past the end of the "DataBlocks" data which indicates that the next data block occurs in a subsequent encapsulating packet.

Having the "BlockOffset" always point at the next available data block allows for recovering the next full inner packet in the presence of outer encapsulating packet loss.

An example IP-TFS packet flow can be found in Appendix A.

2.2.1. Data Blocks

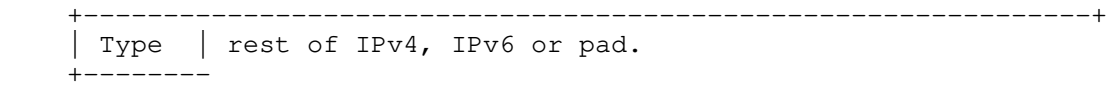


Figure 2: Layout of IP-TFS data block

A data block is defined by a 4-bit type code followed by the data block data. The type values have been carefully chosen to coincide

with the IPv4/IPv6 version field values so that no per-data block type overhead is required to encapsulate an IP packet. Likewise, the length of the data block is extracted from the encapsulated IPv4 or IPv6 packet's length field.

2.2.2. No Implicit End Padding Required

It's worth noting that since a data block type is identified by its first octet there is never a need for an implicit pad at the end of an encapsulating packet. Even when the start of a data block occurs near the end of a encapsulating packet such that there is no room for the length field of the encapsulated header to be included in the current encapsulating packet, the fact that the length comes at a known location and is guaranteed to be present is enough to fetch the length field from the subsequent encapsulating packet payload. Only when there is no data to encapsulated is end padding required, and then an explicit "Pad Data Block" would be used to identify the padding.

2.2.3. Fragmentation, Sequence Numbers and All-Pad Payloads

In order for a receiver to be able to reassemble fragmented inner-packets, the sender MUST send the inner-packet fragments back-to-back in the logical IP-TFS packet stream (i.e., using consecutive ESP sequence numbers). However, the sender is allowed to insert "all-pad" IP-TFS packets (i.e., packets having payloads with a "BlockOffset" of zero and a single pad "DataBlock") in between the IP-TFS packets carrying the inner-packet fragment payloads. This possible interleaving of all-pad packets allows the sender to always be able to send an IP-TFS tunnel packet, regardless of the encapsulation computational requirements.

When a receiver is reassembling an inner-packet, and it receives an "all-pad" IP-TFS tunnel packet, it increments the expected sequence number that the next inner-packet fragment is expected to arrive in.

2.2.4. Empty Payload

In order to support reporting of congestion control information (described later) on a non-IP-TFS enabled SA, IP-TFS allows for the sending of an IP-TFS payload with no data blocks (i.e., the ESP payload length is equal to the IP-TFS header length). This special payload is called an empty payload.

2.2.5. IP Header Value Mapping

[RFC4301] provides some direction on when and how to map various values from an inner IP header to the outer encapsulating header, namely the Don't-Fragment (DF) bit ([RFC0791] and [RFC8200]), the Differentiated Services (DS) field [RFC2474] and the Explicit Congestion Notification (ECN) field [RFC3168]. Unlike [RFC4301], IP-TFS may and often will be encapsulating more than one IP packet per ESP packet. To deal with this, these mappings are restricted further. In particular IP-TFS never maps the inner DF bit as it is unrelated to the IP-TFS tunnel functionality; IP-TFS never IP fragments the inner packets and the inner packets will not affect the fragmentation of the outer encapsulation packets. Likewise, the ECN value need not be mapped as any congestion related to the constant-send-rate IP-TFS tunnel is unrelated (by design!) to the inner traffic flow. Finally, by default the DS field SHOULD NOT be copied although an implementation MAY choose to allow for configuration to override this behavior. An implementation SHOULD also allow the DS value to be set by configuration.

2.3. Exclusive SA Use

It is not the intention of this specification to allow for mixed use of an IP-TFS enabled SA. In other words, an SA that has IP-TFS enabled is exclusively for IP-TFS use and MUST NOT have non-IP-TFS payloads such as IP (IP protocol 4), TCP transport (IP protocol 6), or ESP pad packets (protocol 59) intermixed with non-empty IP-TFS (IP protocol TBD1) payloads. While it's possible to envision making the algorithm work in the presence of sequence number skips in the IP-TFS payload stream, the added complexity is not deemed worthwhile. Other IPsec uses can configure and use their own SAs.

2.4. Modes of Operation

Just as with normal IPsec/ESP tunnels, IP-TFS tunnels are unidirectional. Bidirectional IP-TFS functionality is achieved by setting up 2 IP-TFS tunnels, one in either direction.

An IP-TFS tunnel can operate in 2 modes, a non-congestion controlled mode and congestion controlled mode.

2.4.1. Non-Congestion Controlled Mode

In the non-congestion controlled mode IP-TFS sends fixed-sized packets at a constant rate. The packet send rate is constant and is not automatically adjusted regardless of any network congestion (e.g., packet loss).

For similar reasons as given in [RFC7510] the non-congestion controlled mode should only be used where the user has full administrative control over the path the tunnel will take. This is required so the user can guarantee the bandwidth and also be sure as to not be negatively affecting network congestion [RFC2914]. In this case packet loss should be reported to the administrator (e.g., via syslog, YANG notification, SNMP traps, etc) so that any failures due to a lack of bandwidth can be corrected.

2.4.2. Congestion Controlled Mode

With the congestion controlled mode, IP-TFS adapts to network congestion by lowering the packet send rate to accommodate the congestion, as well as raising the rate when congestion subsides. Since overhead is per packet, by allowing for maximal fixed-size packets and varying the send rate transport overhead is minimized.

The output of the congestion control algorithm will adjust the rate at which the ingress sends packets. While this document does not require a specific congestion control algorithm, best current practice RECOMMENDS that the algorithm conform to [RFC5348]. Congestion control principles are documented in [RFC2914] as well. An example of an implementation of the [RFC5348] algorithm which matches the requirements of IP-TFS (i.e., designed for fixed-size packet and send rate varied based on congestion) is documented in [RFC4342].

The required inputs for the TCP friendly rate control algorithm described in [RFC5348] are the receiver's loss event rate and the sender's estimated round-trip time (RTT). These values are provided by IP-TFS using the congestion information header fields described in Section 3. In particular these values are sufficient to implement the algorithm described in [RFC5348].

At a minimum, the congestion information must be sent, from the receiver and from the sender, at least once per RTT. Prior to establishing an RTT the information SHOULD be sent constantly from the sender and the receiver so that an RTT estimate can be established. The lack of receiving this information over multiple consecutive RTT intervals should be considered a congestion event that causes the sender to adjust it's sending rate lower. For example, [RFC4342] calls this the "no feedback timeout" and it is equal to 4 RTT intervals. When a "no feedback timeout" has occurred [RFC4342] halves the sending rate.

An implementation MAY choose to always include the congestion information in it's IP-TFS payload header if sending on an IP-TFS enabled SA. Since IP-TFS normally will operate with a large packet

size, the congestion information should represent a small portion of the available tunnel bandwidth. An implementation choosing to always send the data MAY also choose to only update the "LossEventRate" and "RTT" header field values it sends every "RTT" though.

When an implementation is choosing a congestion control algorithm (or a selection of algorithms) one should remember that IP-TFS is not providing for reliable delivery of IP traffic, and so per packet ACKs are not required and are not provided.

It's worth noting that the variable send-rate of a congestion controlled IP-TFS tunnel, is not private; however, this send-rate is being driven by network congestion, and as long as the encapsulated (inner) traffic flow shape and timing are not directly affecting the (outer) network congestion, the variations in the tunnel rate will not weaken the provided inner traffic flow confidentiality.

2.4.2.1. Circuit Breakers

In additional to congestion control, implementations MAY choose to define and implement circuit breakers [RFC8084] as a recovery method of last resort. Enabling circuit breakers is also a reason a user may wish to enable congestion information reports even when using the non-congestion controlled mode of operation. The definition of circuit breakers are outside the scope of this document.

3. Congestion Information

In order to support the congestion control mode, the sender needs to know the loss event rate and also be able to approximate the RTT ([RFC5348]). In order to obtain these values the receiver sends congestion control information on it's SA back to the sender. Thus, in order to support congestion control the receiver must have a paired SA back to the sender (this is always the case when the tunnel was created using IKEv2). If the SA back to the sender is a non-IP-TFS enabled SA then an IPTFS_PROTOCOL empty payload (i.e., header only) is used to convey the information.

In order to calculate a loss event rate compatible with [RFC5348], the receiver needs to have a round-trip time estimate. Thus the sender communicates this estimate in the "RTT" header field. On startup this value will be zero as no RTT estimate is yet known.

In order for the sender to estimate it's "RTT" value, the sender places a timestamp value in the "TVal" header field. On first receipt of this "TVal", the receiver records the new "TVal" value along with the time it arrived locally, subsequent receipt of the same "TVal" MUST not update the recorded time. When the receiver

sends it's CC header it places this latest recorded value in the "TEcho" header field, along with 2 delay values, "Echo Delay" and "Transmit Delay". The "Echo Delay" value is the time delta from the recorded arrival time of "TVal" and the current clock in microseconds. The second value, "Transmit Delay", is the receiver's current transmission delay on the tunnel (i.e., the average time between sending packets on it's half of the IP-TFS tunnel). When the sender receives back it's "TVal" in the "TEcho" header field it calculates 2 RTT estimates. The first is the actual delay found by subtracting the "TEcho" value from it's current clock and then subtracting "Echo Delay" as well. The second RTT estimate is found by adding the received "Transmit Delay" header value to the senders own transmission delay (i.e., the average time between sending packets on it's half of the IP-TFS tunnel). The larger of these 2 RTT estimates SHOULD be used as the "RTT" value. The two estimates are required to handle different combinations of faster or slow tunnel packet paths with fast or slow fixed tunnel rates. Choosing the larger of the two values guarantees that the "RTT" is never considered faster than the aggregate transmission delay based on the IP-TFS tunnel rate (the second estimate), as well as never being considered faster than the actual RTT along the tunnel packet path (the first estimate).

The receiver also calculates, and communicates in the "LossEventRate" header field, the loss event rate for use by the sender. This is slightly different from [RFC4342] which periodically sends all the loss interval data back to the sender so that it can do the calculation. See Appendix B for a suggested way to calculate the loss event rate value. Initially this value will be zero (indicating no loss) until enough data has been collected by the receiver to update it.

3.1. ECN Support

In additional to normal packet loss information IP-TFS supports use of the ECN bits in the encapsulating IP header [RFC3168] for identifying congestion. If ECN use is enabled and a packet arrives at the egress endpoint with the Congestion Experienced (CE) value set, then the receiver considers that packet as being dropped, although it does not drop it. The receiver MUST set the E bit in any IPTFS_PROTOCOL payload header containing a "LossEventRate" value derived from a CE value being considered.

As noted in [RFC3168] the ECN bits are not protected by IPsec and thus may constitute a covert channel. For this reason ECN use SHOULD NOT be enabled by default.

4. Configuration

IP-TFS is meant to be deployable with a minimal amount of configuration. All IP-TFS specific configuration should be able to be specified at the unidirectional tunnel ingress (sending) side. It is intended that non-IKEv2 operation is supported, at least, with local static configuration.

4.1. Bandwidth

Bandwidth is a local configuration option. For non-congestion controlled mode the bandwidth SHOULD be configured. For congestion controlled mode one can configure the bandwidth or have no configuration and let congestion control discover the maximum bandwidth available. No standardized configuration method is required.

4.2. Fixed Packet Size

The fixed packet size to be used for the tunnel encapsulation packets can be configured manually or can be automatically determined using Path MTU discovery (see [RFC1191] and [RFC8201]). No standardized configuration method is required.

4.3. Congestion Control

Congestion control is a local configuration option. No standardized configuration method is required.

5. IKEv2

5.1. USE_TFS Notification Message

When using IKEv2, a new "USE_IPTFS" Notification Message is used to enable operation of IP-TFS on a child SA pair. The method used is similar to how USE_TRANSPORT_MODE is negotiated, as described in [RFC7296].

To request IP-TFS operation on the Child SA pair, the initiator includes the USE_IPTFS notification in an SA payload requesting a new Child SA (either during the initial IKE_AUTH or during non-rekeying CREATE_CHILD_SA exchanges). If the request is accepted then response MUST also include a notification of type USE_IPTFS. If the responder declines the request the child SA will be established without IP-TFS enabled. If this is unacceptable to the initiator, the initiator MUST delete the child SA.

The USE_IPTFS notification MUST NOT be sent, and MUST be ignored, during a CREATE_CHILD_SA rekeying exchange as it is not allowed to change IP-TFS operation during rekeying.

The USE_IPTFS notification contains a 1 octet payload of flags that specify any requirements from the sender of the message. If any requirement flags are not understood or cannot be supported by the receiver then the receiver should not enable IP-TFS mode (either by not responding with the USE_IPTFS notification, or in the case of the initiator, by deleting the child SA if the now established non-IP-TFS operation is unacceptable).

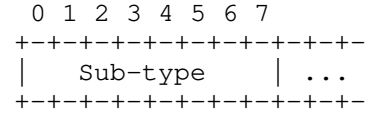
The notification type and payload flag values are defined in Section 6.1.4.

6. Packet and Data Formats

6.1. IP-TFS Payload

ESP Payload Type: 0x5

An IP-TFS payload is identified by the ESP payload type IPTFS_PROTOCOL which has the value 0x5. The first octet of this payload indicates the format of the remaining payload data.

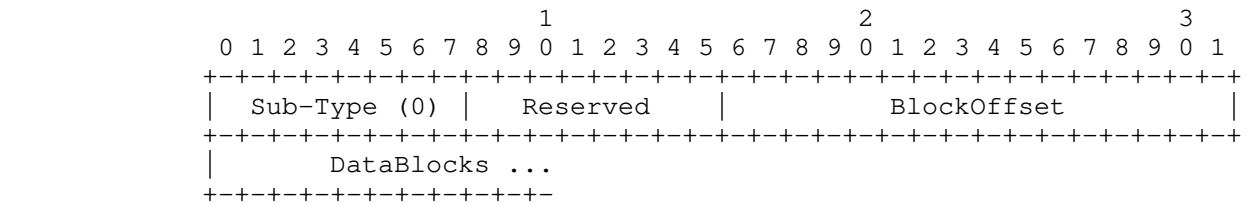


Sub-type:
An 8 bit value indicating the payload format.

This specification defines 2 payload sub-types. These payload formats are defined in the following sections.

6.1.1. Non-Congestion Control IPTFS_PROTOCOL Payload Format

The non-congestion control IPTFS_PROTOCOL payload is comprised of a 4 octet header followed by a variable amount of "DataBlocks" data as shown below.



Sub-type:
: An octet indicating the payload format. For this non-congestion control format, the value is 0.

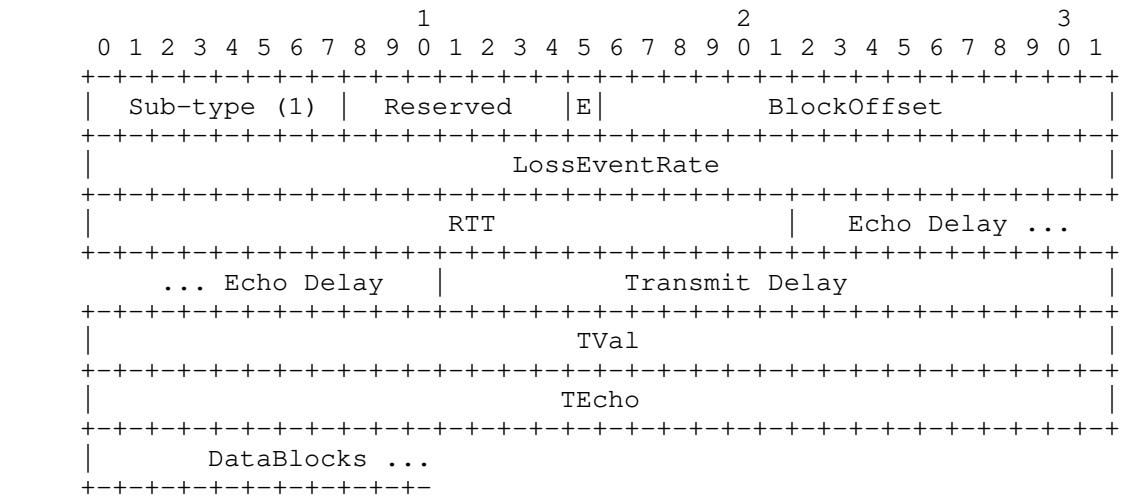
Reserved:
: An octet set to 0 on generation, and ignored on receipt.

BlockOffset:
: A 16 bit unsigned integer counting the number of octets of "DataBlocks" data before the start of a new data block. "BlockOffset" can count past the end of the "DataBlocks" data in which case all the "DataBlocks" data belongs to the previous data block being re-assembled. If the "BlockOffset" extends into subsequent packets it continues to only count subsequent "DataBlocks" data (i.e., it does not count subsequent packets non-"DataBlocks" octets).

DataBlocks:
: Variable number of octets that begins with the start of a data block, or the continuation of a previous data block, followed by zero or more additional data blocks.

6.1.2. Congestion Control IPTFS_PROTOCOL Payload Format

The congestion control IPTFS_PROTOCOL payload is comprised of a 24 octet header followed by a variable amount of "DataBlocks" data as shown below.



Sub-type:
: An octet indicating the payload format. For this congestion control format, the value is 1.

Reserved:
: A 7 bit field set to 0 on generation, and ignored on receipt.

E:
: A 1 bit value if set indicates that Congestion Experienced (CE) ECN bits were received and used in deriving the reported "LossEventRate".

BlockOffset:
: The same value as the non-congestion controlled payload format value.

LossEventRate:
: A 32 bit value specifying the inverse of the current loss event rate as calculated by the receiver. A value of zero indicates no loss. Otherwise the loss event rate is "1/LossEventRate".

RTT:
: A 22 bit value specifying the sender's current round-trip time estimate in microseconds. The value MAY be zero prior to the sender having calculated a round-trip time estimate. The value SHOULD be set to zero on non-IP-TFS enabled SAs. If the value is equal to or larger than "0x3FFFFF" it MUST be set to "0x3FFFFF".

Echo Delay:

A 21 bit value specifying the delay in microseconds incurred between the receiver first receiving the "TVal" value which it is sending back in "TEcho". If the value is equal to or larger than "0x1FFFFF" it MUST be set to "0x1FFFFF".

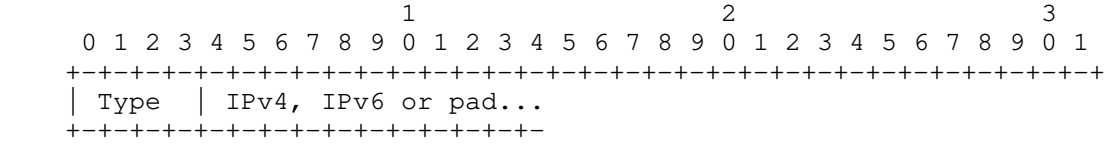
Transmit Delay:
A 21 bit value specifying the transmission delay in microseconds. This is the fixed (or average) delay on the receiver between it sending packets on the IPTFS tunnel. If the value is equal to or larger than "0x1FFFFF" it MUST be set to "0x1FFFFF".

TVal:
An opaque 32 bit value that will be echoed back by the receiver in later packets in the "TEcho" field, along with a "Delay" value of how long that echo took.

TEcho:
The opaque 32 bit value from a received packet's "TVal" field. The received "TVal" is placed in "TEcho" along with a "Delay" value indicating how long it has been since receiving the "TVal" value.

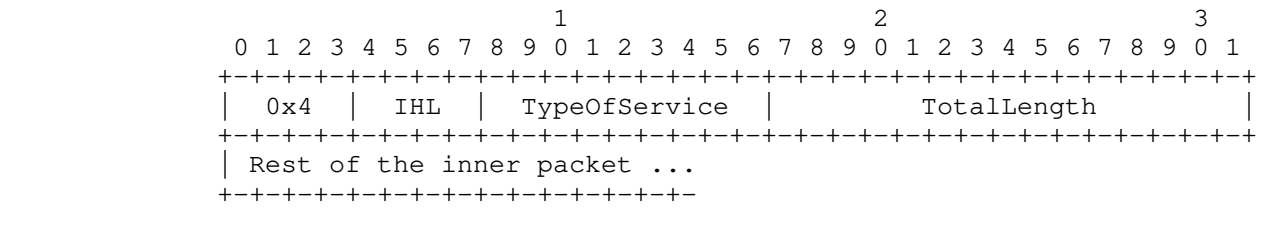
DataBlocks:
Variable number of octets that begins with the start of a data block, or the continuation of a previous data block, followed by zero or more additional data blocks. For the special case of sending congestion control information on an non-IP-TFS enabled SA this value MUST be empty (i.e., be zero octets long).

6.1.3. Data Blocks



Type:
A 4 bit field where 0x0 identifies a pad data block, 0x4 indicates an IPv4 data block, and 0x6 indicates an IPv6 data block.

6.1.3.1. IPv4 Data Block

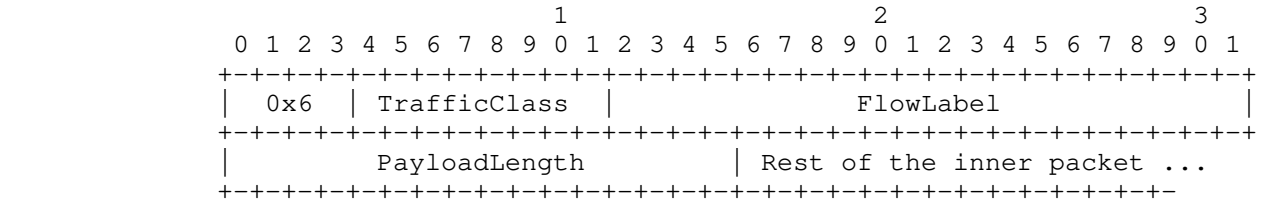


These values are the actual values within the encapsulated IPv4 header. In other words, the start of this data block is the start of the encapsulated IP packet.

Type:
A 4 bit value of 0x4 indicating IPv4 (i.e., first nibble of the IPv4 packet).

TotalLength:
The 16 bit unsigned integer "Total Length" field of the IPv4 inner packet.

6.1.3.2. IPv6 Data Block

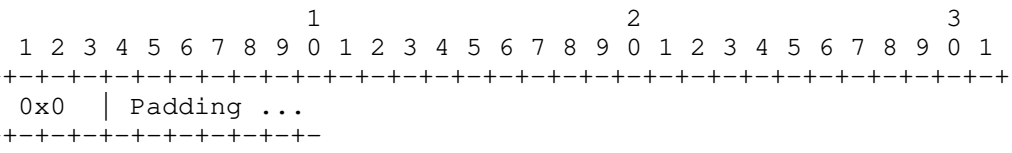


These values are the actual values within the encapsulated IPv6 header. In other words, the start of this data block is the start of the encapsulated IP packet.

Type:
A 4 bit value of 0x6 indicating IPv6 (i.e., first nibble of the IPv6 packet).

PayloadLength:
The 16 bit unsigned integer "Payload Length" field of the inner IPv6 inner packet.

6.1.3.3. Pad Data Block



Type:
A 4 bit value of 0x0 indicating a padding data block.

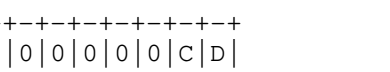
Padding:
extends to end of the encapsulating packet.

6.1.4. IKEv2 USE_IPTFS Notification Message

As discussed in Section 5.1 a notification message USE_IPTFS is used to negotiate IP-TFS operation in IKEv2.

The USE_IPTFS Notification Message State Type is (TBD2).

The notification payload contains 1 octet of requirement flags. There are currently 2 requirement flags defined. This may be revised by later specifications.



0:
6 bits - reserved, MUST be zero on send, unless defined by later specifications.

C:
Congestion Control bit. If set, then the sender is requiring that congestion control information MUST be returned to it periodically as defined in Section 3.

D:
Don’t Fragment bit, if set indicates the sender of the notify message does not support receiving packet fragments (i.e., inner packets MUST be sent using a single "Data Block"). This value only applies to what the sender is capable of receiving; the sender MAY still send packet fragments unless similarly restricted by the receiver in it’s USE_IPTFS notification.

7. IANA Considerations

7.1. IPTFS_PROTOCOL Type

This document requests a protocol number IPTFS_PROTOCOL be allocated by IANA from "Assigned Internet Protocol Numbers" registry for identifying the IP-TFS payload.

Type:
TBD1

Description:
An IP-TFS payload.

Reference:
This document

7.2. IPTFS_PROTOCOL Sub-Type Registry

This document requests IANA create a registry called "IPTFS_PROTOCOL Sub-Type Registry" under "IPTFS_PROTOCOL Parameters" IANA registries. The registration policy for this registry is "Standards Action" ([RFC8126] and [RFC7120]).

Name:
IPTFS_PROTOCOL Sub-Type Registry

Description:
IPTFS_PROTOCOL Payload Formats.

Reference:
This document

This initial content for this registry is as follows:

Sub-Type	Name	Reference
0	Non-Congestion Control Format	This document
1	Congestion Control Format	This document
3-255	Reserved	

7.3. USE_IPTFS Notify Message Status Type

This document requests a status type USE_IPTFS be allocated from the "IKEv2 Notify Message Types - Status Types" registry.

Value:
TBD2

Name:
USE_IPTFS

Reference:
This document

8. Security Considerations

This document describes a mechanism to add Traffic Flow Confidentiality to IP traffic. Use of this mechanism is expected to increase the security of the traffic being transported. Other than the additional security afforded by using this mechanism, IP-TFS utilizes the security protocols [RFC4303] and [RFC7296] and so their security considerations apply to IP-TFS as well.

As noted previously in Section 2.4.2, for TFC to be fully maintained the encapsulated traffic flow should not be affecting network congestion in a predictable way, and if it would be then non-congestion controlled mode use should be considered instead.

9. References

9.1. 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, <https://www.rfc-editor.org/info/rfc2119>.

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

[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, <https://www.rfc-editor.org/info/rfc7296>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

9.2. Informative References

[AppCrypt] Schneier, B., "Applied Cryptography: Protocols, Algorithms, and Source Code in C", 11 2017.

[I-D.iab-wire-image]
Trammell, B. and M. Kuehlewind, "The Wire Image of a Network Protocol", draft-iab-wire-image-01 (work in progress), November 2018.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <https://www.rfc-editor.org/info/rfc1191>.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <https://www.rfc-editor.org/info/rfc2474>.

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <https://www.rfc-editor.org/info/rfc2914>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <https://www.rfc-editor.org/info/rfc3168>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

[RFC4342] Floyd, S., Kohler, E., and J. Padhye, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC)", RFC 4342, DOI 10.17487/RFC4342, March 2006, <https://www.rfc-editor.org/info/rfc4342>.

[RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, DOI 10.17487/RFC5348, September 2008, <https://www.rfc-editor.org/info/rfc5348>.

[RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, DOI 10.17487/RFC7120, January 2014, <https://www.rfc-editor.org/info/rfc7120>.

[RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", RFC 7510, DOI 10.17487/RFC7510, April 2015, <https://www.rfc-editor.org/info/rfc7510>.

[RFC8084] Fairhurst, G., "Network Transport Circuit Breakers", BCP 208, RFC 8084, DOI 10.17487/RFC8084, March 2017, <https://www.rfc-editor.org/info/rfc8084>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <https://www.rfc-editor.org/info/rfc8201>.

Appendix A. Example Of An Encapsulated IP Packet Flow

Below an example inner IP packet flow within the encapsulating tunnel packet stream is shown. Notice how encapsulated IP packets can start and end anywhere, and more than one or less than 1 may occur in a single encapsulating packet.

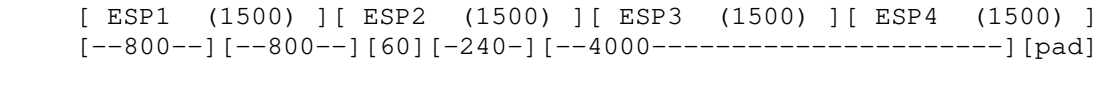


Figure 3: Inner and Outer Packet Flow

The encapsulated IP packet flow (lengths include IP header and payload) is as follows: an 800 octet packet, an 800 octet packet, a 60 octet packet, a 240 octet packet, a 4000 octet packet.

The "BlockOffset" values in the 4 IP-TFS payload headers for this packet flow would thus be: 0, 100, 2900, 1400 respectively. The first encapsulating packet ESP1 has a zero "BlockOffset" which points at the IP data block immediately following the IP-TFS header. The following packet ESP2s "BlockOffset" points inward 100 octets to the start of the 60 octet data block. The third encapsulating packet ESP3 contains the middle portion of the 4000 octet data block so the

offset points past its end and into the forth encapsulating packet. The fourth packet ESP4s offset is 1400 pointing at the padding which follows the completion of the continued 4000 octet packet.

Appendix B. A Send and Loss Event Rate Calculation

The current best practice indicates that congestion control SHOULD be done in a TCP friendly way. A TCP friendly congestion control algorithm is described in [RFC5348]. For this IP-TFS use case (as with [RFC4342]) the (fixed) packet size is used as the segment size for the algorithm. The main formula in the algorithm for the send rate is then as follows:

$$X = \frac{1}{R * (\sqrt{2*p/3} + 12*\sqrt{3*p/8}*p*(1+32*p^2))}$$

Where "X" is the send rate in packets per second, "R" is the round trip time estimate and "p" is the loss event rate (the inverse of which is provided by the receiver).

In addition the algorithm in [RFC5348] also uses an "X_recv" value (the receiver's receive rate). For IP-TFS one MAY set this value according to the sender's current tunnel send-rate ("X").

The IP-TFS receiver, having the RTT estimate from the sender can use the same method as described in [RFC5348] and [RFC4342] to collect the loss intervals and calculate the loss event rate value using the weighted average as indicated. The receiver communicates the inverse of this value back to the sender in the IPTFS_PROTOCOL payload header field "LossEventRate".

The IP-TFS sender now has both the "R" and "p" values and can calculate the correct sending rate. If following [RFC5348] the sender SHOULD also use the slow start mechanism described therein when the IP-TFS SA is first established.

Appendix C. Comparisons of IP-TFS

C.1. Comparing Overhead

C.1.1. IP-TFS Overhead

The overhead of IP-TFS is 40 bytes per outer packet. Therefore the octet overhead per inner packet is 40 divided by the number of outer packets required (fractional allowed). The overhead as a percentage of inner packet size is a constant based on the Outer MTU size.

```
OH = 40 / Outer Payload Size / Inner Packet Size
OH % of Inner Packet Size = 100 * OH / Inner Packet Size
OH % of Inner Packet Size = 4000 / Outer Payload Size
```

Type	IP-TFS	IP-TFS	IP-TFS
MTU	576	1500	9000
PSize	536	1460	8960
40	7.46%	2.74%	0.45%
576	7.46%	2.74%	0.45%
1500	7.46%	2.74%	0.45%
9000	7.46%	2.74%	0.45%

Figure 4: IP-TFS Overhead as Percentage of Inner Packet Size

C.1.2. ESP with Padding Overhead

The overhead per inner packet for constant-send-rate padded ESP (i.e., traditional IPsec TFC) is 36 octets plus any padding, unless fragmentation is required.

When fragmentation of the inner packet is required to fit in the outer IPsec packet, overhead is the number of outer packets required to carry the fragmented inner packet times both the inner IP overhead (20) and the outer packet overhead (36) minus the initial inner IP overhead plus any required tail padding in the last encapsulation packet. The required tail padding is the number of required packets times the difference of the Outer Payload Size and the IP Overhead minus the Inner Payload Size. So:

```
Inner Paylaod Size = IP Packet Size - IP Overhead
Outer Payload Size = MTU - IPsec Overhead

                Inner Payload Size
NF0 = ---------------------------------
       Outer Payload Size - IP Overhead

NF = CEILING(NF0)

OH = NF * (IP Overhead + IPsec Overhead)
     - IP Overhead
     + NF * (Outer Payload Size - IP Overhead)
     - Inner Payload Size

OH = NF * (IPsec Overhead + Outer Payload Size)
     - (IP Overhead + Inner Payload Size)

OH = NF * (IPsec Overhead + Outer Payload Size)
     - Inner Packet Size
```

C.2. Overhead Comparison

The following tables collect the overhead values for some common L3 MTU sizes in order to compare them. The first table is the number of octets of overhead for a given L3 MTU sized packet. The second table is the percentage of overhead in the same MTU sized packet.

Type	ESP+Pad	ESP+Pad	ESP+Pad	IP-TFS	IP-TFS	IP-TFS
L3 MTU	576	1500	9000	576	1500	9000
PSize	540	1464	8964	536	1460	8960
40	500	1424	8924	3.0	1.1	0.2
128	412	1336	8836	9.6	3.5	0.6
256	284	1208	8708	19.1	7.0	1.1
536	4	928	8428	40.0	14.7	2.4
576	576	888	8388	43.0	15.8	2.6
1460	268	4	7504	109.0	40.0	6.5
1500	228	1500	7464	111.9	41.1	6.7
8960	1408	1540	4	668.7	245.5	40.0
9000	1368	1500	9000	671.6	246.6	40.2

Figure 5: Overhead comparison in octets

Type	ESP+Pad	ESP+Pad	ESP+Pad	IP-TFS	IP-TFS	IP-TFS
MTU	576	1500	9000	576	1500	9000
PSize	540	1464	8964	536	1460	8960
40	1250.0%	3560.0%	22310.0%	7.46%	2.74%	0.45%
128	321.9%	1043.8%	6903.1%	7.46%	2.74%	0.45%
256	110.9%	471.9%	3401.6%	7.46%	2.74%	0.45%
536	0.7%	173.1%	1572.4%	7.46%	2.74%	0.45%
576	100.0%	154.2%	1456.2%	7.46%	2.74%	0.45%
1460	18.4%	0.3%	514.0%	7.46%	2.74%	0.45%
1500	15.2%	100.0%	497.6%	7.46%	2.74%	0.45%
8960	15.7%	17.2%	0.0%	7.46%	2.74%	0.45%
9000	15.2%	16.7%	100.0%	7.46%	2.74%	0.45%

Figure 6: Overhead as Percentage of Inner Packet Size

C.3. Comparing Available Bandwidth

Another way to compare the two solutions is to look at the amount of available bandwidth each solution provides. The following sections consider and compare the percentage of available bandwidth. For the sake of providing a well understood baseline normal (unencrypted) Ethernet as well as normal ESP values are included.

C.3.1. Ethernet

In order to calculate the available bandwidth the per packet overhead is calculated first. The total overhead of Ethernet is 14+4 octets of header and CRC plus and additional 20 octets of framing (preamble, start, and inter-packet gap) for a total of 38 octets. Additionally the minimum payload is 46 octets.

Size	E + P	E + P	E + P	IPTFS	IPTFS	IPTFS	Enet	ESP
MTU	590	1514	9014	590	1514	9014	any	any
OH	74	74	74	78	78	78	38	74
40	614	1538	9038	45	42	40	84	114
128	614	1538	9038	146	134	129	166	202
256	614	1538	9038	293	269	258	294	330
536	614	1538	9038	614	564	540	574	610
576	1228	1538	9038	659	606	581	614	650
1460	1842	1538	9038	1672	1538	1472	1498	1534
1500	1842	3076	9038	1718	1580	1513	1538	1574
8960	11052	10766	9038	10263	9438	9038	8998	9034
9000	11052	10766	18076	10309	9480	9078	9038	9074

Figure 7: L2 Octets Per Packet

Size MTU OH	E + P 590 74	E + P 1514 74	E + P 9014 74	IPTFS 590 78	IPTFS 1514 78	IPTFS 9014 78	Enet any 38	ESP any 74
40	2.0M	0.8M	0.1M	27.3M	29.7M	31.0M	14.9M	11.0M
128	2.0M	0.8M	0.1M	8.5M	9.3M	9.7M	7.5M	6.2M
256	2.0M	0.8M	0.1M	4.3M	4.6M	4.8M	4.3M	3.8M
536	2.0M	0.8M	0.1M	2.0M	2.2M	2.3M	2.2M	2.0M
576	1.0M	0.8M	0.1M	1.9M	2.1M	2.2M	2.0M	1.9M
1460	678K	812K	138K	747K	812K	848K	834K	814K
1500	678K	406K	138K	727K	791K	826K	812K	794K
8960	113K	116K	138K	121K	132K	138K	138K	138K
9000	113K	116K	69K	121K	131K	137K	138K	137K

Figure 8: Packets Per Second on 10G Ethernet

Size	E + P 590 74	E + P 1514 74	E + P 9014 74	IPTFS 590 78	IPTFS 1514 78	IPTFS 9014 78	Enet any 38	ESP any 74
40	6.51%	2.60%	0.44%	87.30%	94.93%	99.14%	47.62%	35.09%
128	20.85%	8.32%	1.42%	87.30%	94.93%	99.14%	77.11%	63.37%
256	41.69%	16.64%	2.83%	87.30%	94.93%	99.14%	87.07%	77.58%
536	87.30%	34.85%	5.93%	87.30%	94.93%	99.14%	93.38%	87.87%
576	46.91%	37.45%	6.37%	87.30%	94.93%	99.14%	93.81%	88.62%
1460	79.26%	94.93%	16.15%	87.30%	94.93%	99.14%	97.46%	95.18%
1500	81.43%	48.76%	16.60%	87.30%	94.93%	99.14%	97.53%	95.30%
8960	81.07%	83.22%	99.14%	87.30%	94.93%	99.14%	99.58%	99.18%
9000	81.43%	83.60%	49.79%	87.30%	94.93%	99.14%	99.58%	99.18%

Figure 9: Percentage of Bandwidth on 10G Ethernet

A sometimes unexpected result of using IP-TFS (or any packet aggregating tunnel) is that, for small to medium sized packets, the available bandwidth is actually greater than native Ethernet. This is due to the reduction in Ethernet framing overhead. This increased bandwidth is paid for with an increase in latency. This latency is the time to send the unrelated octets in the outer tunnel frame. The following table illustrates the latency for some common values on a 10G Ethernet link. The table also includes latency introduced by padding if using ESP with padding.

	ESP+Pad 1500	ESP+Pad 9000	IP-TFS 1500	IP-TFS 9000
40	1.14 us	7.14 us	1.17 us	7.17 us
128	1.07 us	7.07 us	1.10 us	7.10 us
256	0.97 us	6.97 us	1.00 us	7.00 us
536	0.74 us	6.74 us	0.77 us	6.77 us
576	0.71 us	6.71 us	0.74 us	6.74 us
1460	0.00 us	6.00 us	0.04 us	6.04 us
1500	1.20 us	5.97 us	0.00 us	6.00 us

Figure 10: Added Latency

Notice that the latency values are very similar between the two solutions; however, whereas IP-TFS provides for constant high bandwidth, in some cases even exceeding native Ethernet, ESP with padding often greatly reduces available bandwidth.

Appendix D. Acknowledgements

We would like to thank Don Fedyk for help in reviewing and editing this work.

Appendix E. Contributors

The following people made significant contributions to this document.

Lou Berger
LabN Consulting, L.L.C.

Email: lberger@labn.net

Author's Address

Christian Hopps
LabN Consulting, L.L.C.

Email: chopps@chopps.org

Network
Internet-Draft
Updates: 7296 (if approved)
Intended status: Standards Track
Expires: May 3, 2021

P. Wouters
S. Prasad
Red Hat

October 30, 2020

Labeled IPsec Traffic Selector support for IKEv2
draft-ietf-ipsecme-labeled-ipsec-04

Abstract

This document defines a new Traffic Selector (TS) Type for Internet Key Exchange version 2 to add support for negotiating Mandatory Access Control (MAC) security labels as a traffic selector of the Security Policy Database (SPD). Security Labels for IPsec are also known as "Labeled IPsec". The new TS type is TS_SECLABEL, which consists of a variable length opaque field specifying the security label. This document updates the IKEv2 TS negotiation specified in RFC 7296 Section 2.9.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 3, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

In computer security, Mandatory Access Control usually refers to systems in which all subjects and objects are assigned a security label. A security label is comprised of a set of security attributes. The security labels along with a system authorization policy determine access. Rules within the system authorization policy determine whether the access will be granted based on the security attributes of the subject and object.

Traditionally, security labels used by Multilevel Systems (MLS) are comprised of a sensitivity level (or classification) field and a compartment (or category) field, as defined in [FIPS188] and [RFC5570]. As MAC systems evolved, other MAC models gained in popularity. For example, SELinux, a Flux Advanced Security Kernel (FLASK) implementation, has security labels represented as colon-separated ASCII strings composed of values for identity, role, and type. The security labels are often referred to as security contexts.

Traffic Selector (TS) payloads specify the selection criteria for packets that will be forwarded over the newly set up IPsec SA as enforced by the Security Policy Database (SPD, see [RFC4301]). This document updates the Traffic Selector negotiation specified in Section 2.9 of [RFC7296].

This document specifies a new Traffic Selector Type TS_SECLABEL for IKEv2 that can be used to negotiate security labels as additional selectors for the Security Policy Database (SPD) to further restrict the type of traffic allowed to be sent and received over the IPsec SA.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Traffic Selector clarification

The negotiation of Traffic Selectors is specified in Section 2.9 of [RFC7296] where it defines two TS Types (TS_IPV4_ADDR_RANGE and TS_IPV6_ADDR_RANGE). The Traffic Selector payload format is specified in Section 3.13 of [RFC7296]. However, the term Traffic Selector is used to denote the traffic selector payloads and individual traffic selectors of that payload. Sometimes the exact meaning can only be learned from context or if the item is written in plural ("Traffic Selectors" or "TSs"). This section clarifies these terms as follows:

A Traffic Selector (no acronym) is one selector for traffic of a specific Traffic Selector Type (TS_TYPE). For example a Traffic Selector of TS_TYPE TS_IPV4_ADDR_RANGE for UDP traffic in the IP network 198.51.100.0/24 covering all ports, is denoted as (17, 0, 198.51.100.0-198.51.100.255)

A Traffic Selector payload (TS) is a set of one or more Traffic Selectors of the same or different TS_TYPEs, but MUST include at least one TS_TYPE of TS_IPV4_ADDR_RANGE or TS_IPV6_ADDR_RANGE. For example, the above Traffic Selector by itself in a TS payload is denoted as TS((17, 0, 198.51.100.0-198.51.100.255))

1.3. Traffic Selector update

The negotiation of Traffic Selectors is specified in Section 2.9 of [RFC7296] and states that the TSi/TSr payloads MUST contain at least one Traffic Selector type. This document updates the text to mean that the TSi/TSr payloads MUST contain at least one Traffic Selector of type TS_IPV4_ADDR_RANGE or TS_IPV6_ADDR_RANGE, as other Traffic Selector types can be defined that are complimentary to these Traffic Selector Types and cannot be selected on their own without TS_IPV4_ADDR_RANGE or TS_IPV6_ADDR_RANGE. The below defined TS_SECLABEL Traffic Selector Type is an example of this.

2. TS_SECLABEL Traffic Selector Type

This document defines a new TS Type, TS_SECLABEL that contains a single new opaque Security Label.

2.1. TS_SECLABEL payload format

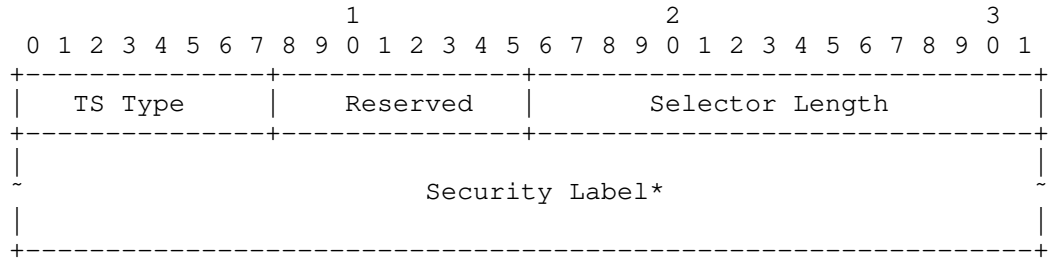


Figure 1: Labeled IPsec Traffic Selector

*Note: All fields other than TS Type and Selector Length depend on the TS Type. The fields shown is for TS Type TS_SECLABEL, the selector this document defines.

- TS Type (one octet) - Set to [TBD] for TS_SECLABEL,
- Selector Length (2 octets, unsigned integer) - Specifies the length of this Traffic Selector substructure including the header.
- Security Label - An opaque byte stream of at least one octet.

2.2. TS_SECLABEL properties

The TS_SECLABEL Traffic Selector Type does not support narrowing or wildcards. It MUST be used as an exact match value.

If the TS_SECLABEL is present in a TSi/TSr, at least one Traffic Selector of type TS_IPV4_ADDR_RANGE or TS_IPV6_ADDR_RANGE MUST also be present in that TSi/TSr.

The Security Label contents are opaque to the IKE implementation. That is, the IKE implementation might not have any knowledge of the meaning of this selector, other than as a type and opaque value to pass to the SPD.

A zero length Security Label MUST NOT be used. If a received TS payload contains a TS_TYPE of TS_SECLABEL with a zero length Security Label, that specific Traffic Selector MUST be ignored. If no other Traffic Selector of TS_TYPE TS_SECLABEL can be selected, a TS_UNACCEPTABLE Error Notify message MUST be returned. A zero length Security Label MUST NOT be interpreted as a wildcard security label.

If multiple Security Labels are allowed for a given IP protocol, start and end address/port match, multiple TS_SECLABEL can be included in a TS payload.

If the Security Label traffic selector is optional from a configuration point of view, the initiator will have to choose which TS payload to attempt first. If it includes the Security Label and receives a TS_UNACCEPTABLE, it can attempt a new Child SA negotiation without that Security Label.

A responder that selected a TS with TS_SECLABEL MUST use the Security Label for all selector operations on the resulting IPsec SA. It MUST NOT select a TS_set with a TS_SECLABEL without using the specified Security Label, even if it deems the Security Label optional, as the initiator TS_set with TS_SECLABEL means the initiator mandates using that Security Label.

3. Traffic Selector negotiation

This document updates the [RFC7296] specification as follows:

Each TS payload (TSi and TSr) MUST contain at least one TS_TYPE of TS_IPV4_ADDR_RANGE or TS_IPV6_ADDR_RANGE.

Each TS payload (TSi or TSr) MAY contain one or more other TS_TYPEs, such as TS_SECLABEL.

A responder MUST create its TS response by selecting one of each TS_TYPE present in the offered TS by the initiator. If it cannot select one of each TS_TYPE, it MUST return a TS_UNACCEPTABLE Error Notify payload.

If a specific TS_TYPE (other than TS_IPV4_ADDR_RANGE or TS_IPV6_ADDR_RANGE which are mandatory) is deemed optional, the initiator SHOULD first try to negotiate the Child SA with the TS payload including the optional TS_TYPE. Upon receiving TS_UNACCEPTABLE, it SHOULD attempt a new Child SA negotiation using the same TS but without the optional TS_TYPE.

Some TS_TYPE's support narrowing, where the responder is allowed to select a subset of the original TS. Narrowing MUST NOT result in an empty selector for that TS_TYPE.

3.1. Example TS negotiation

An initiator could send:

```
TSi = ((17,0,192.0.2.0-192.0.2.255),
       (0,0,198.51.0-198.51.255),
       TS_SECLABEL1, TS_SECLABEL2)

TSr = ((17,0,203.0.113.0-203.0.113.255),
       (0,0,203.0.113.0-203.0.113.255),
       TS_SECLABEL1, TS_SECLABEL2)
```

Figure 2: initiator TS payloads example

The responder could answer with the following example:

```
TSi = ((0,0,198.51.0-198.51.255),
       TS_SECLABEL1)

TSr = (((0,0,203.0.113.0-203.0.113.255),
       TS_SECLABEL1)
```

Figure 3: responder TS payloads example

3.2. Considerations for using multiple TS_TYPEs in a TS

It would be unlikely that the traffic for TSi and TSr would have a different Security Label, but this specification does allow this to be specified. If the initiator does not support this, and wants to prevent the responder from picking different labels for the TSi / TSr payloads, it should attempt a Child SA negotiation with only the first Security Label first, and upon failure retry a new Child SA negotiation with only the second Security Label.

If different IP ranges can only use different specific Security Labels, than these should be negotiated in two different Child SA negotiations. If in the example above, the initiator only allows 192.0.2.0/24 with TS_SECLABEL1, and 198.51.0/24 with TS_SECLABEL2, than it MUST NOT combine these two ranges and security labels into one Child SA negotiation.

The mechanism of narrowing of Traffic Selectors with TS_IPV4_ADDR_RANGE and TS_IPV6_ADDR_RANGE does not apply to TS_SECLABEL as the Security Label itself is not interpreted and cannot itself be narrowed. It MUST be matched exactly. Rekey of an IPsec SA MUST only use identical Traffic Selectors, which means the same TS Type and selectors MUST be used. This guarantees that a Security Label once negotiated, remains part of the IPsec SA after a rekey.

4. Security Considerations

It is assumed that the Security Label can be matched by the IKE implementation to its own configured value, even if the IKE implementation itself cannot interpret the Security Label value.

A packet that matches an SPD entry for all components except the Security Label would be treated as "not matching". If no other SPD entries match, the (mis-labeled) traffic might end up being transmitted in the clear. It is presumed that other Mandatory Access Control methods are in place to prevent mis-labeled traffic from reaching the IPsec subsystem, or that the IPsec subsystem itself would install a REJECT/DISCARD rule in the SPD to prevent unlabeled traffic otherwise matching a labeled security SPD rule from being transmitted without IPsec protection.

5. IANA Considerations

This document defines two new entries in the IKEv2 Traffic Selector Types registry:

Value	TS Type	Reference
TBD	TS_SECLABEL	[this document]

Figure 4

6. Implementation Status

[Note to RFC Editor: Please remove this section and the reference to [RFC6982] before publication.]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

Authors are requested to add a note to the RFC Editor at the top of this section, advising the Editor to remove the entire section before publication, as well as the reference to [RFC7942].

6.1. Libreswan

Organization: The Libreswan Project

Name: https://lists.libreswan.org/mailman/listinfo/swan-dev/

Description: A Proof of Concept branch is available for interop testing.

Level of maturity: Alpha

Coverage: Implements the entire draft using SElinux based labels

Licensing: GPLv2

Implementation experience: TBD

Contact: Libreswan Development: swan-dev@libreswan.org

7. Acknowledgements

A large part of the introduction text was taken verbatim from [draft-jml-ipsec-ikev2-security-label] whose authors are J Latten, D. Quigley and J. Lu.

8. References

8.1. 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, <https://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, <https://www.rfc-editor.org/info/rfc7296>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

8.2. Informative References

[draft-jml-ipsec-ikev2-security-label] Latten, J., Quigley, D., and J. Lu, "Security Label Extension to IKE", draft-wouters-edns-tcp-keeaplive (work in progress), January 2011.

[FIPS188] NIST, "National Institute of Standards and Technology, "Standard Security Label for Information Transfer"", Federal Information Processing Standard (FIPS) Publication 188, September 1994.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

[RFC5570] StJohns, M., Atkinson, R., and G. Thomas, "Common Architecture Label IPv6 Security Option (CALIPSO)", RFC 5570, DOI 10.17487/RFC5570, July 2009, <https://www.rfc-editor.org/info/rfc5570>.

[RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, DOI 10.17487/RFC6982, July 2013, <https://www.rfc-editor.org/info/rfc6982>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

Authors' Addresses

Paul Wouters
Red Hat

Email: pwouters@redhat.com

Sahana Prasad
Red Hat

Email: sahana@redhat.com

Network
Internet-Draft
Updates: 7296,8221,8247 (if approved)
Intended status: Standards Track
Expires: January 14, 2021

P. Wouters, Ed.
Red Hat

July 13, 2020

Deprecation of IKEv1 and obsoleted algorithms
draft-pwouters-ikev1-ipsec-graveyard-05

Abstract

Internet Key Exchange version 1 (IKEv1) is deprecated. Accordingly, IKEv1 has been moved to Historic status. A number of old algorithms that are associated with IKEv1, and not widely implemented for IKEv2 are deprecated as well. IANA is instructed to close all IKEv1 registries.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 14, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

IKEv1 [RFC2409] and its related documents for ISAKMP [RFC2408] and IPsec DOI [RFC2407] were obsoleted by IKEv2 [RFC4306] in December 2005. The latest version of IKEv2 at the time of writing was published in 2014 in [RFC7296]. The Internet Key Exchange (IKE) version 2 has replaced version 1 over 15 years ago. IKEv2 has now seen wide deployment and provides a full replacement for all IKEv1 functionality. No new modifications or new algorithms have been accepted for IKEv1 for at least a decade. IKEv2 addresses various issues present in IKEv1, such as IKEv1 being vulnerable to amplification attacks. IKEv1 has been moved to Historic status, and this document requests IANA to close all IKEv1 registries.

Algorithm implementation requirements and usage guidelines for IKEv2 [RFC8247] and ESP/AH [RFC8223] gives guidance to implementors but limits that guidance to avoid broken or weak algorithms. It does not deprecate algorithms that have aged and are not in use, but leave these algorithms in a state of "MAY be used". This document deprecates those algorithms that are no longer advised but for which there are no known attacks resulting in their earlier deprecation.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. RFC 2409 to Historic

IKEv1 is deprecated. Systems running IKEv1 should be upgraded and reconfigured to run IKEv2. Systems that support IKEv1 but not IKEv2 are most likely also unsuitable candidates for continued operation. Such unsupported systems have a much higher chance of containing an implementation vulnerability that will never be patched. IKEv1 systems can be abused for packet amplification attacks. IKEv1 systems most likely do not support modern algorithms such as AES-GCM or CHACHA20_POLY1305 and quite often only support or have been configured to use the very weak Diffie-Hellman Groups 2 and 5. IKEv1 systems should be upgraded or replaced by IKEv2 systems.

IKEv1 and its way of using Preshared Keys (PSKs) protects against quantum computer based attacks. IKEv2 updated its use of PSK to improve the error reporting, but at the expense of post-quantum security. If post-quantum security is required, these systems should be migrated to use IKEv2 Postquantum Preshared Keys (PPK) [RFC8784]

Some IKEv1 implementations support Labeled IPsec, a method to negotiate an addition Security Context selector to the SPD, but this method was never standarized in IKEv1. Those IKEv1 systems that require Labeled IPsec should migrate to an IKEv2 system supporting Labeled IPsec as specified in [draft-ietf-ipsecme-labeled-ipsec].

4. Deprecating obsolete algorithms

This document deprecates the following algorithms:

- Encryption Algorithms: RC5, IDEA, CAST, Blowfish, and the unspecified 3IDEA, ENCR_DES_IV64 and ENCR_DES_IV32
- PRF Algorithms: the unspecified PRF_HMAC_TIGER
- Integrity Algorithms: HMAC-MD5-128
- Diffie-Hellman groups: none

5. Security Considerations

There are only security benefits by deprecating IKEv1 for IKEv2.

The deprecated algorithms have long been in disuse and are no longer actively deployed or researched. It presents an unknown security risk that is best avoided. Additionally, these algorithms not being supported in implementations simplifies those implementations and reduces the accidental use of these deprecated algorithms through misconfiguration or downgrade attacks.

6. IANA Considerations

This document instructs IANA to mark all IKEv1 registries as DEPRECATED.

Additionally, this document instructs IANA to add an additional Status column to the IKEv2 Transform Type registries and mark the following entries as DEPRECATED:

Transform Type 1 - Encryption Algorithm IDs

Number	Name	Status
1	ENCR_DES_IV64	DEPRECATED [this document]
2	ENCR_DES	DEPRECATED [RFC8247]
4	ENCR_RC5	DEPRECATED [this document]
5	ENCR_IDEA	DEPRECATED [this document]
6	ENCR_CAST	DEPRECATED [this document]
7	ENCR_BLOWFISH	DEPRECATED [this document]
8	ENCR_3IDEA	DEPRECATED [this document]
9	ENCR_DES_IV32	DEPRECATED [this document]

Figure 1

Transform Type 2 - Pseudorandom Function Transform IDs

Number	Name	Status
1	PRF_HMAC_MD5	DEPRECATED [RFC8247]
1	PRF_HMAC_TIGER	DEPRECATED [this document]

Figure 2

Transform Type 3 - Integrity Algorithm Transform IDs

Number	Name	Status
1	AUTH_HMAC_MD5_96	DEPRECATED [RFC8247]
3	AUTH_DES_MAC	DEPRECATED [RFC8247]
4	AUTH_KPDK_MD5	DEPRECATED [RFC8247]
6	AUTH_HMAC_MD5_128	DEPRECATED [this document]
7	AUTH_HMAC_SHA1_160	DEPRECATED [this document]

Figure 3

Transform Type 4 - Diffie Hellman Group Transform IDs

Number	Name	Status
1	768-bit MODP Group	DEPRECATED [RFC8247]
22	1024-bit MODP Group with 160-bit Prime Order Subgroup	DEPRECATED [RFC8247]

Figure 4

All entries not mentioned here should receive no value in the new Status field.

This document instructs IANA to close and mark as obsolete the Internet Key Exchange (IKE) Attributes registries as well as the "Magic Numbers" for ISAKMP Protocol registries.

The IESG is requested to designate IKEv1 to Historic.

7. References

7.1. 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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2407] Piper, D., "The Internet IP Security Domain of Interpretation for ISAKMP", RFC 2407, DOI 10.17487/RFC2407, November 1998, <https://www.rfc-editor.org/info/rfc2407>.

[RFC2408] Maughan, D., Schertler, M., Schneider, M., and J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, DOI 10.17487/RFC2408, November 1998, <https://www.rfc-editor.org/info/rfc2408>.

[RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, DOI 10.17487/RFC2409, November 1998, <https://www.rfc-editor.org/info/rfc2409>.

[RFC4306] Kaufman, C., Ed., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, DOI 10.17487/RFC4306, December 2005, <https://www.rfc-editor.org/info/rfc4306>.

[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, <https://www.rfc-editor.org/info/rfc7296>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8223] Esale, S., Torvi, R., Jalil, L., Chunduri, U., and K. Raza, "Application-Aware Targeted LDP", RFC 8223, DOI 10.17487/RFC8223, August 2017, <https://www.rfc-editor.org/info/rfc8223>.

[RFC8247] Nir, Y., Kivinen, T., Wouters, P., and D. Migault, "Algorithm Implementation Requirements and Usage Guidance for the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 8247, DOI 10.17487/RFC8247, September 2017, <https://www.rfc-editor.org/info/rfc8247>.

[RFC8784] Fluhrer, S., Kampanakis, P., McGrew, D., and V. Smyslov, "Mixing Preshared Keys in the Internet Key Exchange Protocol Version 2 (IKEv2) for Post-quantum Security", RFC 8784, DOI 10.17487/RFC8784, June 2020, <https://www.rfc-editor.org/info/rfc8784>.

7.2. Informative References

[draft-ietf-ipsecme-labeled-ipsec] Wouters, P. and S. Prasad, "Labeled IPsec Traffic Selector support for IKEv2", draft-ietf-ipsecme-labeled-ipsec (work in progress), March 2019.

Author's Address

Paul Wouters (editor)
Red Hat

Email: pwouters@redhat.com

Network
Internet-Draft
Intended status: Standards Track
Expires: May 6, 2021

A. Antony
S. Klassert
secunet
P. Wouters
Red Hat

November 2, 2020

IKEv2 support for per-queue Child SAs
draft-pwouters-multi-sa-performance-00

Abstract

This document defines two Notification Payload (NUM_QUEUES and QUEUE_INFO) for the Internet Key Exchange Protocol Version 2 (IKEv2). These payloads add support for negotiating multiple identical Child SAs that can be used to to optimize performance based on the number of queues or CPUs, orcw to create multiple Child SAs for different Quality of Service (QoS) levels.

Using multiple identical Child Sa's has the additional benefit that multiple streams have their own Sequence Number, ensuring that CPU's don't have to synchronize their crypto state or disable their replay window detection.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 6, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

IPsec implementations are currently limited to using one queue or CPU per Child SA. The result is that a machine with many queues/CPUs is limited to only using one these per Child SA. This severely limits the speeds that can be obtained. An unencrypted link of 10gbps or more is commonly reduced to 2-3gbps when IPsec is used to encrypt the link, for example when using AES-GCM.

Furthermore IPsec implementations are currently limited to use the same Child SA for all Quality of Service (QoS) types bacause the QoS type is not a part of the TS. The result is that IPsec can't do active Quality of Service priorizing without disabling the anti replay detection.

To make better use of multiple network queues and CPUs, it can be beneficial to negotiate and install multiple identical Child SAs. IKEv2 [RFC7296] already allows installing multiple identical Child SAs, but often implementations will assume the older Child SA is being replaced by the newer Child Sa, even when no INITIAL_CONTACT notify payload was received.

When two IKEv2 peers want to negotiate multiple Child SAs, it would be useful for them to convey how many of these are considered acceptable to install. This avoids triggering CREATE_CHILD_SA exchanges that will be rejected with TS_UNACCEPTABLE.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Performance bottlenecks

Currently, most IPsec implementations are limited by using one CPU or network queue per Child SA. There are a number of performance reasons for this, but a key limitation is that sharing the AEAD state, counters and sequence numbers between multiple CPUs is not feasible without a significant performance penalty. There is a need to negotiate and establish multiple Child SA's with identical TSi/TSr on a per-queue or per-CPU basis.

3. Negotiation of performance specific Child SAs

The number of Child SA's notify payload refers to the number of instances for this particular TSi/TSr combination. Both ends send their Preferred number of Child SAs and the maximum of Child SAs they are willing to install. Both ends pick the highest preferred number up to the lowest maximum number. That is if one end prefers 16 but accepts 32, and the other end prefers 48 and accepts 48, the number picked is 32. If a 33rd Child SA is attempted, the peer with the 32 maximum SHOULD return TS_UNACCEPTABLE.

The NUM_QUEUES Notify is sent as part of the IKE_AUTH or CREATE_CHILD_SA message that contains the Traffic Selector payload for a new Child SA. If there are multiple IKE_AUTH exchanges, such as when using EAP, the TSi/TSr payloads and the Notify payloads defines in this document only appear in the first IKE_AUTH message. In CREATE_CHILD_SA, the NUM_QUEUES Notify MUST only be sent in

messages for new set of Child SA's (the message used to set up the Head SA)

The QUEUE_INFO Notify MUST only be sent in CREATE_CHILD_SA for Sub SA's. During CREATE_CHILD_SA's sent for Child SA rekey, the QUEUE_INFO MAY be included. If it is included it MUST be the same as for the Child SA being rekeyed.

4. Implementation specifics

There are various considerations that an implementation could use to determine the best way to install the multiple Child SAs. Below are examples of such strategies.

4.1. One Child per CPU

A simple distribution could be to install one Child SA per CPU. Note that at least one of the Child SAs must be the "fallback" in case there is no specific Child SA on a specific CPU. This is called the Head SA, where the per-CPU Child SA is called a Sub CA. The initial Child SA negotiated with IKE becomes the Head SA. This ensures that any CPU generating traffic to be encrypted has an available (if not optimal) Child SA to use. Any subsequent Child SA's with identical TSi/TSr are considered Sub SA's and installed to be used only by a single CPU.

Implementations supporting per-CPU SAs SHOULD extend their mechanism of on-demand negotiation that is triggered by traffic to include a CPU (or queue) identifier in their ACQUIRE message from the SPD to the IKE daemon (eg via NETLINK of PFKEYv2). If the kernel's ACQUIRE message does not support sending a per-CPU identifier, then the IKE daemon should initiate all its Child SAs immediately upon receiving an ACQUIRE.

Performing per-CPU Child SA negotiations can result in both peers initiating Sub SAs at once. This is especially likely in the per-CPU acquire case. Responders should install the Sub SA on the CPU with the least amount of Sub SA's for this TSi/TSr pair. It should count outstanding ACQUIREs as an assigned Sub SA. It is still possible that when the peers only have one slot left to assign, that both peers send an ACQUIRE at the same time. The initiator that receives the CREATE_CHID_SA response last, eg the initiator of the slowest duplicate MAY send a delete to delete the duplicate Child SA.

As an optimization, Sub SA's that see little traffic MAY be deleted. However, it MUST NOT delete an idle Head SA. This ensures both peers always have a Child SA that can be used by a CPU that does not have a

Sub SA (yet) and ensures encrypted traffic can always be exchanged, even when that traffic triggered a new per-CPU ACQUIRE.

When the number of queues or CPUs are different between the peers, the peer with the least amount of queues or CPUs MAY decide to not install a second outbound Child SA as it will never use it to send traffic. However, it MUST install all inbound Child SA's as it cannot predict which of these the other peer will use to send traffic. It MUST NOT generate an error when deleting the (missing) outbound SA component of the Child SA.

A per-CPU ACQUIRE message SHOULD still send the Traffic Selector (TSi) information of the trigger packet. This information MAY be used by the responder to select the most efficient target CPU to use. For example, if the trigger packet was for TCP destination port 25 (SMTP), it might be able to install the Child SA on the CPU that is also running the mail server process. See [RFC7296] Section 2.9.

The QUEUE_INFO Notify payload MAY be sent in the CREATE_CHILD_SA request for the additional (subSA) Child SAs. It can be used to convey the QoS stream or CPUID.

[Clarify narrowing Traffic Selectors. Should it be allowed/forbidden ?]

[Clarify CP / INTERNAL_ADDRESS. Should it be allowed/forbidden ?]

[UDP enacap Due to the nature handling of UDP encapsulated ESP at the receiver NIC queus and intermediate routers for parallel paths, UDP encapsulated ESP will used multiple source ports. NOTE: this is implemented in libreswan on Linux XFRM.]

4.2. QoS Child SA's

To install multiple Child SA's for different QoS levels, a method similar to per-CPU is used. The initial Child SA is used for all QoS levels not matched by more specific Child SA's. Additional Child SA's are installed per QoS level, which can be done on-demand if the kernel's IPsec subsystem can send per-QoS level ACQUIREs to the IKE daemon.

A request for a Child SA for a specific QoS value MUST include the QUEUE_INFO Notify payload set to the required QoS value so that both endpoints use the same Child SA for the same QoS level. If a certain QoS level proposed is not acceptable to the resonder, TS_UNACCEPTABLE MUST be returned. During Child SA REKEY, the QUEUE_INFO Notify MAY be included but MUST contain the same value as the Child SA that is

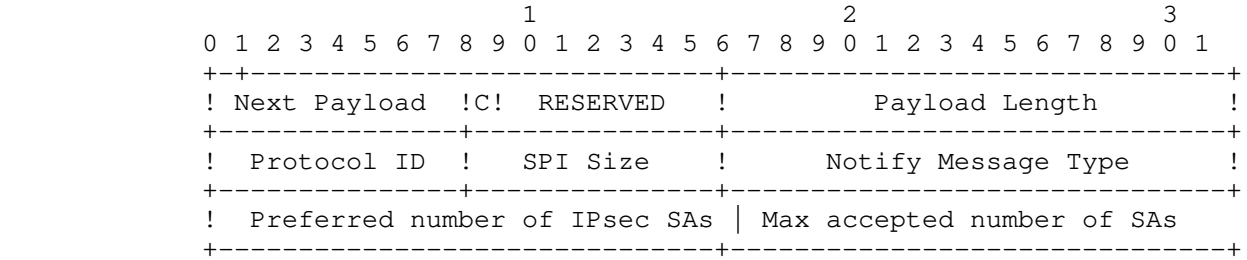
being rekeyed. [This kind of suggests this should be a TS_TYPE and not a Notify]

5. Payload Format

All multi-octet fields representing integers are laid out in big endian order (also known as "most significant byte first", or "network byte order").

5.1. NUM_QUEUES Notify Payload

The NUM_QUEUES Notify payload is related to a Child SA, and MAY be exchanged in IKE_AUTH or in a CREATE_CHILD_SA for new SA. It MUST NOT be sent in CREATE_CHILD_SA for REKEY. If received for a REKEY operation, it MUST be ignored. See [RFC7296] Section 1.3.1.



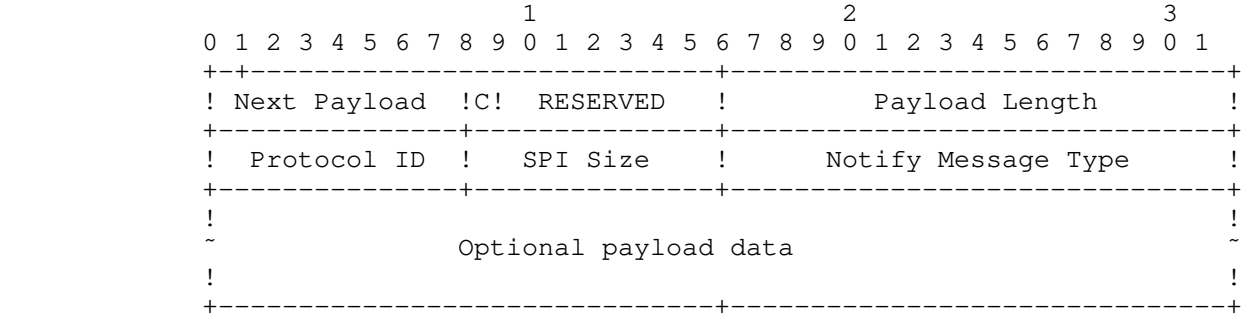
- Protocol ID (1 octet) - MUST be 0. MUST be ignored if not 0.
- SPI Size (1 octet) - MUST be 0. MUST be ignored if not 0. by the IPsec protocol ID
- Notify Message Type (2 octets) - set to [TBD]
- Preferred number of per-CPU IPsec SAs (2 octets). Value MUST be greater than 0. If 0 is received, it MUST be interpreted as 1.
- Maximum accepted number of per-CPU IPsec SAs (2 octets). Value MUST be greater than 0. If 0 is received, it MUST be interpreted as 1.

Note: The first Child SA that is not bound to a single CPU (Head SA) is not counted as part of these numbers.

5.2. QUEUE_INFO Notify Payload

The QUEUE_INFO Notify payload is an optional related to a Child SA, and MAY be exchanged in IKE_AUTH or in a CREATE_CHILD_SA for new SA.

It MUST NOT be sent in CREATE_CHILD_SA for REKEY, see [RFC7296] Section 1.3.1.



- Protocol ID (1 octet) - MUST be 0. MUST be ignored if not 0.
- SPI Size (1 octet) - MUST be 0. MUST be ignored if not 0. by the IPsec protocol ID
- Notify Message Type (2 octets) - set to [TBD]
- Optional Payload Data. This can be to identify the QoS options or CPU-ID [Probable needs to be specified by this document]

6. Security Considerations

[TO DO]

7. Implementation Status

[Note to RFC Editor: Please remove this section and the reference to [RFC6982] before publication.]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

Authors are requested to add a note to the RFC Editor at the top of this section, advising the Editor to remove the entire section before publication, as well as the reference to [RFC7942].

7.1. Linux XFRM

Organization: Linux kernel XFRM

Name: XFRM-PCPU-v1
https://git.kernel.org/pub/scm/linux/kernel/git/klassert/linux-stk.git/log/?h=xfrm-pcpu-v1

Description: An initial Kernel IPsec implementation of the per-CPU method.

Level of maturity: Alpha

Coverage: Fully implements Head SA and per-CPU Sub SA's

Licensing: GPLv2

Implementation experience: TBD

Contact: Linux IPsec: members@linux-ipsec.org

7.2. Libreswan

Organization: The Libreswan Project

Name: pcpu-3 https://libreswan.org/wiki/XFRM_pCPU

Description: An initial IKE implementation of the per-CPU method.

Level of maturity: Alpha

Coverage: implements Head SA and per-CPU Sub SA's.

Licensing: GPLv2

Implementation experience: TBD

Contact: Libreswan Development: swan-dev@libreswan.org

7.3. strongSWAN

Organization: Secunet

Name: XXXX https://secunet.com/somethingU

Description: An initial IKE implementation of the per-CPU method.

Level of maturity: Alpha

Coverage: implements Head SA and per-CPU Sub SA's.

Licensing: GPLv2

Implementation experience: TBD

Contact: Antony Antony: antony.antony@secunet.com.

8. IANA Considerations

This document defines one new IKEv2 Notify Message for the IANA "IKEv2 Notify Message Types - Status Types" registry.

Value	Notify Messages - Status Types	Reference
[TBD]	NUM_QUEUES	[this document]
[TBD]	QUEUE_INFO	[this document]

Figure 1

9. References

9.1. 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, <https://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, <https://www.rfc-editor.org/info/rfc7296>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

9.2. Informative References

[RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, DOI 10.17487/RFC6982, July 2013, <https://www.rfc-editor.org/info/rfc6982>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

Authors' Addresses

Antony Antony
secunet Security Networks AG

Email: antony.antony@secunet.com

Steffen Klassert
secunet Security Networks AG

Email: steffen.klassert@secunet.com

Paul Wouters
Red Hat

Email: pwouters@redhat.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 1, 2021

V. Smyslov
ELVIS-PLUS

October 28, 2020

Revised Cookie Processing in the IKEv2 Protocol
draft-smyslov-ipsecme-ikev2-cookie-revised-00

Abstract

This document defines a revised processing of cookies in the Internet Key Exchange protocol Version 2 (IKEv2). It is intended to solve a problem in IKEv2 when due to packets loss and reordering peers may erroneously fail to authenticate each other when cookies are used in the initial IKEv2 exchange.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 1, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

The Internet Key Exchange protocol Version 2 (IKEv2) described in [RFC7296] includes mechanism to defend against DoS attacks. The mechanism is based on cookie which a responder can request an initiator to return in a subsequent request. This allows the responder avoid creating state until it is sure that the initiator's IP address is not spoofed. The cookie mechanism is optional and it is up to the responder whether to use it or not.

When cookie mechanism is used in networks with high probability of packets loss and reordering, it is possible that peers end up with different views on whether cookies were used or not or which content the used cookie had. Since cookie, if used, is a part of an IKEv2 message that is included into calculation of authentication data by both peers, the different views leads to the situation when peers erroneously fail to authenticate each other.

This specification revises processing of cookies in IKEv2 in such a way that peers supporting it exclude cookies from data to be authenticated. This allows them to complete authentication even in the situation described above.

2. Terminology and Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Using Cookie in IKEv2

Section 2.6 of [RFC7296] specifies that when a responder detects a large number of half-open IKE SAs, it SHOULD reply to an IKE_SA_INIT request with a response containing the COOKIE notification and If an IKE_SA_INIT response includes the COOKIE notification, the initiator MUST then retry the IKE_SA_INIT request, and include the COOKIE notification containing the received data as the very first payload in it, retaining all other payloads intact. This is illustrated in the Figure 1.

```
Initiator                              Responder
-----------                            -----------
send req1:
HDR, SAi1, KEi, Ni          -->        recv req1, send resp1:
                            <--        HDR, N(COOKIE,c)
recv resp1, send req2:
HDR, N(COOKIE,c),
     SAi1, KEi, Ni          -->        recv req2, send resp2:
                            <--        HDR, SAr1, KEr, Nr
recv resp2
```

Figure 1

Note, that the responder saves no state when it sends message resp1. This is achieved by the way cookies are generated. A good way to generate a cookie is described in Section 2.6 of [RFC7296]:

Cookie = <VersionIDofSecret> | Hash(Ni | IPi | SPIi | <secret>)

where <secret> is a randomly generated secret known only to the responder and periodically changed. [RFC7296] advises the responder to change the value of <secret> frequently, especially if under attack.

Later in the IKE_AUTH exchange the IKE_SA_INIT messages are authenticated by including their content intact into the data that is signed (or MAC'ed) using peers' credentials (see Section 2.15 of [RFC7296] for details).

4. Problem Description

To successfully complete authentication it is important that both peers use the same content of the IKE_SA_INIT messages when calculating authentication data. However, when cookie is employed, the IKE_SA_INIT request is sent at least twice with different content. Section 2.15. of [RFC7296] states that if the first message of the IKE_SA_INIT exchange is sent multiple times with different

content (such as with a cookie), it is the latest version of the message that is used for authentication. However, in situations when network packets can be lost and reordered peers may end up with different views on what is "the latest version of the message". Two examples of such situations are shown below.

Consider a situation when at the time the initiator starts creating IKE SA by sending req1 message the responder thinks it's under attack and responds with a resp2 message containing cookie request. However, this message is delayed in the network.

```
Initiator                              Responder
-----------                            -----------
send req1:
HDR, SAi1, KEi, Ni        -->          recv req1, send resp1:
 (delayed)                <--          HDR, N(COOKIE,c)
```

Since the initiator hasn't received any response, it retransmits its initial request message req1 after some time. During this time the situation on the responder has changed and it doesn't think it's under attack anymore, so it responds with resp2 message and considers the IKE_SA_INIT exchange completed. This message is also delayed in the network.

```
Initiator                              Responder
-----------                            -----------
re-send req1:
HDR, SAi1, KEi, Ni        -->          recv req1, send resp2:
 (delayed)                <--          HDR, SAr1, KEr, Nr
```

After some time the initiator eventually receives the first initiator's response resp1, which contains cookie request. It is the first response the initiator receives, so it re-sends the request adding the received cookie into it (req2). However, this message is lost and never reaches the responder. Shortly after sending req2 the initiator receives resp2 message.

```
Initiator                              Responder
-----------                            -----------
recv resp1, send req2:
HDR, N(COOKIE,c),
     SAi1, KEi, Ni        -->           (lost)
recv resp2
```

At this point both peers have completed the IKE_SA_INIT exchange and the KE_AUTH exchange is ready to start. However, the peers have different opinions on what the latest IKE_SA_INIT request message was - the initiator thinks it was req2, while the responder thinks it was

req1. As a result - the authentication in the IKE_AUTH exchange will fail.

Let's consider another possible sequence, that leads to the same result. As with the previous example the initiator starts creating IKE SA by sending req1 message. The responder thinks it's under attack and responds with a resp2 message containing cookie request with cookie c1. However, this message is delayed in the network.

```
Initiator                              Responder
-----------                            -----------
send req1:
HDR, SAi1, KEi, Ni        -->          recv req1, send resp1:
 (delayed)                <--          HDR, N(COOKIE, c1)
```

As with the first example, the initiator hasn't received any response and retransmits its initial request message req1 after some time. It happens that within this time the value of <secret> has changed (note, that [RFC7296] advises to do it frequently, especially when under attack). Since the responder cannot verify cookie c1 and it still thinks it is under attack, it acts as if req1 contains no cookie and sends back resp2 message also containing cookie request, with a new cookie c2 (that was calculated using the same input data and the new value of <secret>).

```
Initiator                              Responder
-----------                            -----------
re-send req1:
HDR, SAi1, KEi, Ni        -->          recv req1, send resp2:
                          <--          HDR, N(COOKIE, c2)
```

This message is not delayed, it reaches the initiator and the initiator sends a new request req2 containing cookie c2. The responder receives this message and successfully verifies the cookie using its current value of <secret>. Since the cookie verification is successfull, the responder sends back resp3 message and considers the IKE_SA_INIT exchange completed. However, resp3 message is delayed.

```
Initiator                              Responder
-----------                            -----------
recv resp2, send req2:
HDR, N(COOKIE, c2),
     SAi1, KEi, Ni        -->          recv req2, send resp3:
 (delayed)                <--          HDR, SAr1, KEr, Nr
```

After some time the initiator eventually receives resp1 message, which was delayed. This message contains another value of cookie -

c1. Since this message was received later than resp2 message, the initiator thinks the value c1 is fresher than c2 and sends a new request message req3 now with c1 cookie. This message is lost in the network and never reaches the responder. Shortly after sending req3 the initiator receives resp3 message.

```
Initiator                              Responder
-----------                            -----------
recv resp1, send req3:
HDR, N(COOKIE, c1),
     SAi1, KEi, Ni        -->           (lost)
recv resp3
```

As with the first example, at this point both peers have completed the IKE_SA_INIT exchange and are ready for the KE_AUTH exchange. However, the peers again have different opinions on what the latest IKE_SA_INIT request message was - the initiator thinks it was req3, while the responder thinks it was req2. As a result - the authentication in the IKE_AUTH exchange will fail as with the previous example.

The root of this problem is that the initial request can be re-sent several times with different content depending on the responder's current state, which can change over time. Note, that this situation is generally not possible with the INVALID_KE_PAYLOAD notification, even that in this case the request is also sent several times. This is because the responder will always either require changing Key Exchange method or not, so it is not possible that eventually peers end up with different opinions on what Key Exchange method was negotiated in the IKE_SA_INIT exchange.

5. Revised Cookie Processing

This specification proposes to solve the problem by excluding cookie content from data to be authenticated. The rationale for this is that cookie must be verified by the responder independently at the time it is received in the IKE_SA_INIT request, so there is no need to authenticate it.

5.1. Negotiation of Revised Cookie Processing

For the purpose of using revised cookie processing a new Status Type notify REVISED_COOKIE is defined. Its Notify Message Type is <TBA by IANA>, Protocol ID and SPI Size are both set to 0. The responder includes an empty REVISED_COOKIE notification whenever it sends a response containing COOKIE notification. If the initiator doesn't support this extension it will ignore this notification and continues as described in [RFC7296]. In case the initiator supports revised

cookie processing it will re-send its initial request including the received cookie, but placing the cookie data into the REVISED_COOKIE notification instead of COOKIE notification.

```
Initiator                              Responder
-----------                            -----------
send req1:
HDR, SAi1, KEi, Ni          -->        recv req1, send resp1:
                                       HDR, N(COOKIE,c),
                            <--             N(REVISED_COOKIE)
recv resp1, send req2:
HDR, N(REVISED_COOKIE,c),
     SAi1, KEi, Ni          -->        recv req2, send resp2:
                            <--        HDR, SAr1, KEr, Nr
recv resp2
```

Figure 2

5.2. Processing of REVISED_COOKIE Notification

If the responder has sent the REVISED_COOKIE notification in the message requesting cookie it should be prepared to receive the re-sent IKE_SA_INIT request with the REVISED_COOKIE notification containing the cookie and with no COOKIE notification. Processing of the REVISED_COOKIE notification by the responder MUST be identical to the processing of COOKIE notification which is described in Sections 2.6 and 2.7 of [RFC7296].

5.3. Changes in AUTH Payload Calculation

In the subsequent IKE_AUTH exchange peers authenticate each other by signing (or MAC'ing) blobs of data. These blobs are defined in Section 2.15 of [RFC7296]. In particular, initiator's blob is defined as follows:

```
InitiatorSignedOctets = RealMessage1 | NonceRData | MACedIDForI
GenIKEHDR = [ four octets 0 if using port 4500 ] | RealIKEHDR
RealIKEHDR =  SPIi | SPIr |  . . . | Length
RealMessage1 = RealIKEHDR | RestOfMessage1
NonceRPayload = PayloadHeader | NonceRData
InitiatorIDPayload = PayloadHeader | RestOfInitIDPayload
RestOfInitIDPayload = IDType | RESERVED | InitIDData
MACedIDForI = prf(SK_pi, RestOfInitIDPayload)
```

Figure 3

In Figure 3 RealMessage1 is the latest version of the IKE_SA_INIT request message.

If the very first payload in RealMessage1 is the REVISED_COOKIE notify, then InitiatorSignedOctets are computed as shown in Figure 4. In particular:

1. The content of the REVISED_COOKIE notify payload is eliminated from the message;

2. The Next Payload field in the IKE Header is set to the value of the Next Payload field in the header of eliminated payload;

3. The length of the eliminated payload (indicated in the Length field in its header) is subtracted from the Length field in the IKE Header.

```
InitiatorSignedOctets = PseudoMessage1 | NonceRData | MACedIDForI
RealMessage1 = RealIKEHDR | NotifyREVISED_COOKIE | RestOfMessage1
NotifyREVISED_COOKIE = NextPld | 0 | PldLength | RestOfNotifyCOOKIE
GenIKEHDR = [ four octets 0 if using port 4500 ] | RealIKEHDR
RealIKEHDR = SPIi | SPIr |  HdrNextPld | . . . | MsgLength
PseudoKEHDR =  SPIi | SPIr |  NewHdrNextPld | . . . | NewMsgLength
NewHdrNextPld = NextPld
NewMsgLength = MsgLength - PldLength
PseudoMessage1 = PseudoIKEHDR | RestOfMessage1
NonceRPayload = PayloadHeader | NonceRData
InitiatorIDPayload = PayloadHeader | RestOfInitIDPayload
RestOfInitIDPayload = IDType | RESERVED | InitIDData
MACedIDForI = prf(SK_pi, RestOfInitIDPayload)
```

Figure 4

In brief, if RealMessage1 doesn't contain the REVISED_COOKIE notification then it is used in the authentication as is (Figure 3). Otherwise a new pseudo message PseudoMessage1 is used which is constructed from RealMessage1 as if it doesn't contain the REVISED_COOKIE notification (Figure 4>).

This modification excludes Notify payload containing cookie from the input to the AUTH payload calculation, thus solving the problem described in Section 4.

6. Security Considerations

This extension modifies the way IKE initiator is authenticated to the IKE responder. In particular, the cookie, created by the responder and returned by the initiator in the IKE_SA_INIT request is excluded from the data to be authenticated. IKEv2 specification requires that cookie (if present in the request) be verified by the responder at the early stage of the IKE_SA_INIT request message processing. If

this verification fails, then the responder must act as if no cookie were present (see Section 2.6 of [RFC7296]), which in most cases results in requesting a new cookie. An adversary that is able to modify cookie content (or remove it from the request) will get no new advantages if this extension is used: either the responder requests a new cookie, or it doesn't care about the cookie at the moment and the IKE_SA_INIT exchange succeeded with invalid cookie. In the later case if revised cookie processing is used the subsequent IKE_AUTH exchange will also succeed and IKE SA will be created, which is different from the current situation, when authentication will fail in the IKE_AUTH if cookie is changed by the attacker.

Excluding cookie from the data to be authenticated doesn't degrade security properties of IKEv2, because the content of the cookie is generated by the responder and must be verified by the responder well before the authentication takes place. The initiator doesn't participate in generation of cookie, it only returns it back as a blob.

Compared to the current processing of cookie the difference caused by the revised processing in a situation when an attacker changes cookie in the IKE_SA_INIT request is that IKE SA will still be created (provided no other obstacles exists), but only if the responder at the moment doesn't care about validity of the received cookie (it means that it is not under attack).

7. IANA Considerations

This document defines a new Notify Message Types in the "Notify Message Types - Status Types" registry:

<TBA>	REVISED_COOKIE

8. Acknowledgements

Author is grateful to Tero Kivinen and Wei Pan for sharing their thoughts about this problem and potential ways to solve it.

9. 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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[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, <https://www.rfc-editor.org/info/rfc7296>.

Author's Address

Valery Smyslov
ELVIS-PLUS
PO Box 81
Moscow (Zelenograd) 124460
RU

Phone: +7 495 276 0211
Email: svan@elvis.ru

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 5, 2021

CJ. Tjhai
Post-Quantum
T. Heider
genua GmbH
V. Smyslov
ELVIS-PLUS

November 1, 2020

Beyond 64KB Limit of IKEv2 Payload
draft-tjhai-ikev2-beyond-64k-limit-00

Abstract

The maximum Internet Key Exchange Version 2 (IKEv2) payload size is limited to 64KB. This makes IKEv2 not usable for conservative post-quantum cryptosystem whose public-key is larger than 64KB. This document describes the mechanisms and considerations to exchange such large key-establishment data in IKEv2.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 5, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

Our digital communications are secured by public-key cryptography algorithms that relies on computational hardness assumptions such as the difficulty in factoring large integers or that of finding the discrete logarithm on an elliptic curve group or finite-field. Recent advances in quantum computing, however, have caused some concerns on the security of these assumptions. It is conjectured that these hard computational problems can be solved in polynomial time when sufficiently large quantum computers become available. The concerns have prompted the National Institute of Standards and Technology (NIST) to initiate a process to standardize one or more public-key algorithms that are quantum-resistant. This family of algorithms is known as post-quantum or quantum-resistant cryptographic algorithms.

It would be ideal if these cryptographic algorithms can be drop-in replacements to the classical algorithms we currently use. Unfortunately, almost all of the post-quantum cryptography algorithms have either public-key, ciphertext or signature size that is many times larger than their classical counterparts. One of the issues that this will cause, in particular for UDP-based protocols such as IPsec, is fragmentation of packets at IP layer. In the context of IPsec/IKEv2 post-quantum key exchange, the fragmentation issue can be addressed by sending the post-quantum exchange data in IKE_INTERMEDIATE [I-D.ietf-ipsecme-ikev2-intermediate], which is the

intermediary state between IKE_SA_INIT and IKE_AUTH. This is the approach taken in [I-D.ietf-ipsecme-ikev2-multiple-ke] whereby a classical and one or more post-quantum key exchanges are combined in order to establish security associations that are quantum-resistant.

Because all public-key cryptography algorithms rely on computational hardness assumptions, the confidence of a cryptographic algorithm is an important consideration. An algorithm that has been well-studied and field-tested is better trusted than newer ones. Unfortunately, the confidence of post-quantum cryptographic algorithms is relatively low. All of the algorithms submitted to NIST post-quantum standardization are based on new computational hardness assumptions and despite being conjectured to be resistant to quantum computer attacks, they have not been well cryptanalyzed compared to the classical counterparts. An exception to this is the Goppa-code based McEliece cryptosystem [McEliece] which has withstood years of cryptanalysis since 1978 and still remains unbroken. It is not surprising that a more efficient and CCA secure version of McEliece cryptosystem, Classic McEliece [CM], is selected as one of the finalists in NIST post-quantum standardization. Furthermore, this cryptosystem has also been recommended for long-term confidentiality protection of data, see [BSI].

While there is interest in using McEliece cryptosystem, in particular for information that needs to remain secure for a long time, there is a challenge in integrating it with IKEv2 [RFC7296]. One characteristic of McElieces cryptosystem is the very asymmetric size of its ciphertext and public-key. While its ciphertext is the smallest compared to all other post-quantum key-establishment algorithms submitted to NIST, the size of its public-key, however, is the largest. The smallest public-key size of Classic McEliece is 255KB. This presents a problem if one were to use Classic McEliece for key-establishment with IKEv2, as the maximum payload size supported by IKEv2 is limited to 64KB. This document describes a mechanism to support IKEv2 key-exchange with key size larger than 64KB, building on the works in [I-D.ietf-ipsecme-ikev2-multiple-ke] and [I-D.ietf-ipsecme-ikev2-intermediate].

1.1. Terminology

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].

This document assumes familiarity with IKEv2 concept described in [RFC7296].

2. Fragmentation of Large Payload

A method to extend IKEv2 that allows quantum-resistant shared secret to be derived from at least one post-quantum key-establishment algorithm is proposed in [I-D.ietf-ipsecme-ikev2-multiple-ke]. Each post-quantum key-establishment data is transported using an IKE_INTERMEDIATE message, which appears following an IKE_SA_INIT exchange. This is necessary because most post-quantum key-establishment data are larger than 1KB and therefore are likely to be dropped by firewalls and network middleboxes if they are sent in the IKE_SA_INIT message over a UDP channel. IKEv2 has a mechanism to handle IP-level fragmentation [RFC7383], but it is only available to messages sent after the IKE_SA_INIT exchange. As such, it is necessary to send these post-quantum key-establishment payloads in IKE_INTERMEDIATE so that it can benefit from the IKEv2 message fragmentation mechanism.

IKEv2 message fragmentation [RFC7383] allows a big payload to be broken up into a number of smaller UDP datagrams. However, this mechanism can only be used to fragment payloads up to 64KB in size. For larger messages, different mechanisms are required and two of which are discussed below.

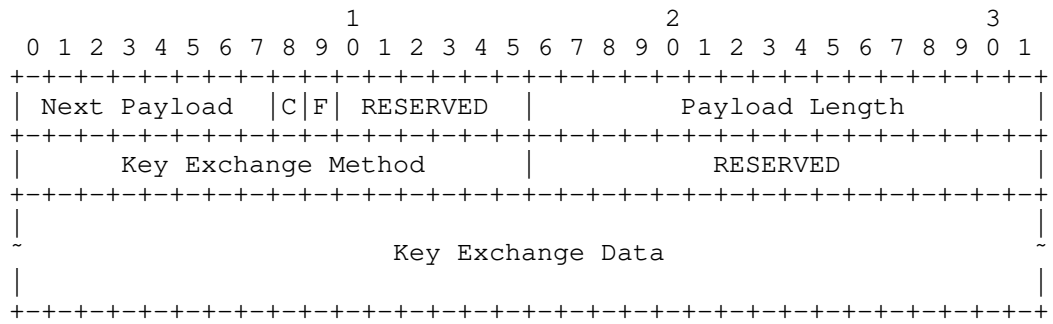
2.1. Hash and URL

[RFC7296] defines a mechanism whereby an authentication payload such as a certificate can be encoded using a hash value and a URL. A peer utilizes HTTP_CERT_LOOKUP_SUPPORTED Notify payload to indicate that X.509 certificates are not transported in-band, instead the other peer shall fetch the certificates from the given URL and verify its integrity from the hash value. In this way, the peer needs to send 20 octets plus a variable length URL only over the wire, instead of a few kilobytes of payload, which is useful in the event IKEv2 message fragmentation is not available.

Large public keys can be transported by reusing the same technique and this can be done in two ways, as described below.

2.1.1. Key Exchange Payload

The Key Exchange Data field of IKEv2 Key Exchange Payload contains a single format, which is a blob that is only meaningful to the specified key exchange method. In order to support hash and URL data, an encoding format needs to be specified on the header.



The reserved bit-field F above specifies the encoding format. If it is 0, the Key Exchange Data is a blob as specified in RFC7296. On the other hand if it is 1, the Key Exchange Data is in the form of hash and URL format, whereby the hash value is the SHA3-256 digest [FIPS-202] of the replaced value truncated to 20 octets and the URL value is a variable length URL (in either http or https schema) that resolves to the DER-encoded of the replaced value itself.

Because the hash and URL value is transported in a Key Exchange Payload, it is possible to support the use-case of a single post-quantum key-establishment with large public-key. This payload will be sent as part of IKE_SA_INIT exchange and it will not require IKE_INTERMEDIATE exchanges.

2.1.2. Certificate Payload

An alternative is to re-purpose Certificate Payload to carry the hash and URL value of the post-quantum key-establishment data. At the time of writing, the IANA registry defines two hash and URL encoding values, namely X.509 certificate and X.509 certificate bundle. In order to use this payload, a new encoding value for key establishment data will be required.

Because a Certificate Payload is part of IKE_AUTH message, unlike the previous approach, the hash and URL value of the key-establishment data shall be transported via IKE_INTERMEDIATE message. As such, it will not be able to support a single post-quantum key-establishment with a large public-key case. Furthermore, it is semantically incorrect to repurpose Certificate Payload, which is intended to carry authentication data, to transport key-establishment data.

2.2. Payload Fragmentation

As mentioned earlier, IKEv2 support for fragmentation as specified in [RFC7383] allows IKE messages up to 64KB to be broken down into smaller fragments. While the size of IKE payloads is constrained to

a 16-bit field, an IKE message itself has a 32-bit length field and therefore, in order to support payloads larger than 64KB limit, a fragmentation needs to be carried out at an upper level. In this case, the large key-establishment data is first fragmented into a number of payload fragments that are no bigger than 64KB and each of which is subjected to further fragmentation using IKEv2 standard message fragmentation when transported in IKE_INTERMEDIATE messages.

There are two possible ways to perform large payload fragmentation, as discussed below.

2.2.1. Bulk Transfer and Confirmation

The large key-establishment payload is first split into a sequence of Key Exchange payload chunks where each share the same value of Key Exchange Method value and has no more than 64KB in size. This sequence of payload chunks is encrypted and is treated as an Encrypted and Authenticated payload as per Section 3.14 of [RFC7296], which is then sent over an IKE_INTERMEDIATE message.

```
 Initiator                       Responder
 -----------------------------------------------------------------
 HDR, SAi1, KEi1, Ni,
   N(IKEV2_FRAGMENTATION_SUPPORTED)*,
   N(INTERMEDIATE_EXCHANGE_SUPPORTED) --->

                                 HDR, SAr1, KEr1, Nr,
                                   N(IKEV2_FRAGMENTATION_SUPPORTED)*,
                           <---    N(INTERMEDIATE_EXCHANGE_SUPPORTED)

 HDR, SK{KEi2.1, KEi2.2, KEi2.3, ...} --->

                           <---  HDR, SK{KEr2, ...}
 *: optional
```

While the IKE header (HDR) has a 32-bit field to denote the length of its message, that for Encrypted payload header (SK) is capped at 16-bit, which is not long enough. As a consequence, the payload length field of the Encrypted Payload SHALL be set to 0. Because the Encrypted payload is always the last payload in an IKE message, it is possible to compute the encrypted IKE payload length instead of relying on the information in its payload length field.

When IKEv2 standard message fragmentation is used, each of the Key Exchange payload chunk is subjected to further fragmentation as per [RFC7383].

2.2.2. Incremental Transfer and Confirmation

As stated in Section 4 of [RFC7383], if any single fragment is lost, the receiving peer will not be able to reassemble the original large key-establishment payload. The above bulk transfer is susceptible to this issue. There is another way to transfer these payload chunks that is less susceptible to this, but at the cost of higher latency. Instead of transferring in a bulk, each Key Exchange payload chunk must be acknowledged prior to sending the subsequent chunk. As before, the large key-establishment payload is split over several Key Exchange payload chunks where each of them share the same Key Exchange Method value. Each chunk is then sent to the peer using the IKE_INTERMEDIATE message, and each one must be acknowledged by the receiving peer before the subsequent chunk can be sent.

```
 Initiator                       Responder
 -----------------------------------------------------------------
 HDR, SAi1, KEi1, Ni,
   N(IKEV2_FRAGMENTATION_SUPPORTED)*,
   N(INTERMEDIATE_EXCHANGE_SUPPORTED) --->

                                 HDR, SAr1, KEr1, Nr,
                                   N(IKEV2_FRAGMENTATION_SUPPORTED)*,
                           <---    N(INTERMEDIATE_EXCHANGE_SUPPORTED)

 HDR, SK{KEi2.1, ...}      --->

                           <---  HDR, SK{}

 HDR, SK{KEi2.2, ...}      --->

                           <---  HDR, SK{}

 HDR, SK{KEi2.3, ...}      --->

                           <---  HDR, SK{KEr2, ...}

 HDR, SK{}                 --->

 *: optional
```

In order to support key-encapsulation mechanism, the receiving peer has to wait until the entire chunks are received before it can respond with its own Key Exchange payload, which may not be large.

While the description above focuses on the transfer of Key Exchange payload larger than 64KB, the same approach can be used to transfer large public-key, certificate or signature if there is a need to

support hybrid authentication or even multiple certificates with large constituent component.

3. Operational Considerations

While using hash and URL method to transport large key-establishment data requires minimal modification to IKEv2 protocol, there are disadvantages from deployment point of view that may make this method impractical. Firstly, an IKE peer that originates a hash and URL value will also need to implement additional infrastructure so that it can serve HTTP requests in order to allow the actual key-establishment data to be fetched. While this may not be an issue for Internet facing peers, in the context of road-warrior or remote-access cases, the hash and URL value is initiated by an IKE peer that is usually a device sitting behind a network address translation (NAT) device and as such, it may not be able to run a publicly reachable HTTP server infrastructure on the same device. An possible solution for these cases is to publish the key-establishment data to a separate server, which is not practical as one cannot expect an IKE initiator to always have deployed an HTTP server. Lastly, IKE peers are predominantly deployed at the network edge where strict firewall rules are generally enforced. The need to open up another port to serve HTTP requests may cause either technical or policy complication that render this approach unacceptable.

Unlike the aforementioned hash and URL method, the payload fragmentation method does not require additional infrastructure, however, there is non-zero probability of lost packets when sending a large number of fragments over a UDP connection. Given a set of fragments, when transmitted, each one of them is not individually acknowledged and if any one of them is lost, the entire set will have to be retransmitted. As a consequence, given the size of the payload and also the potential of multiple retransmissions, there may be a noticeable delay in establishing an security association (SA), in particular in lossy network conditions. Therefore, implementations MAY use a longer timeout value for the purpose of dead-peer detection, but a balance needs to be struck as too large of a value will open up security vulnerabilities as discussed in the following section. In the unlikely event where there is a frequent retransmission due to loss of fragments, implementations MAY send the IKE messages over a TCP connection as specified in [RFC8229]. In this instance, the peers SHOULD NOT advertise support for IKE fragmentation as this is already handled inherently by the TCP stream.

4. Security Considerations

The hash and URL approach is vulnerable to (distributed) denial of service attacks as an unauthenticated rogue peer may trick a legitimate peer to fetch a large amount of random meaningless data from a remote server. Implementations SHOULD NOT blindly download all of the data in the given URL. Because a legitimate key-establishment payload should be DER-encoded, they SHOULD download the first few octets to determine the length of the ASN.1 structure representing these octets, then only continue to download the remaining decoded number of octets if the length is expected for the chosen key-establishment algorithm. It should be noted that the content of the data to be downloaded may be under attacker's control and therefore even if the length is as expected, the content may be meaningless bit that is of no use for key-establishment.

There is no exception to the payload fragmentation method, it is also vulnerable to the same attack vectors. Malicious peers may send a large number of fragments, but incomplete, to the legitimate peer causing memory exhaustion.

In order to counter these attacks, downloading or accepting the transfer of a large number of octets SHOULD only be carried out when the peer has been authenticated. In other words, key-establishment using large data should not be done to establish an IKE SA, but it should only be used to establish Child SA or rekeying of IKE SA from Child SA. If, for whatever reason, an IKE SA has to be established using the large key-establishment data, then it is RECOMMENDED that the strategies and recommendations described in [RFC8019] be implemented.

If TCP encapsulation is used, refer to the security considerations in [RFC8229].

Lastly, downloading or transferring large amounts of data is an expensive operation, bandwidth and memory wise. Consequently, implementations should consider using a longer rekeying interval or SHOULD consider relaxing forward secrecy requirements but using CCA-secure key-establishment algorithm only. With chosen-ciphertext attack (CCA)-secure schemes, there is no loss in security if the public-key is reused.

5. References

5.1. Normative References

[I-D.ietf-ipsecme-ikev2-intermediate]
Smyslov, V., "Intermediate Exchange in the IKEv2 Protocol", draft-ietf-ipsecme-ikev2-intermediate-05 (work in progress), September 2020.

[I-D.ietf-ipsecme-ikev2-multiple-ke]
Tjhai, C., Tomlinson, M., Bartlett, G., Fluhrer, S., Geest, D., Garcia-Morchon, O., and V. Smyslov, "Multiple Key Exchanges in IKEv2", draft-ietf-ipsecme-ikev2-multiple-ke-01 (work in progress), July 2020.

[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, <https://www.rfc-editor.org/info/rfc7296>.

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

5.2. Informative References

[BSI] Federal Office for Information Security, "Cryptographic Mechanisms: Recommendations and Key Lengths", 2020, <https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf>.

[CM] Classic McEliece submission team, "Classic McEliece: NIST post-quantum cryptography standardization finalist", 2020, <https://classic.mceliece.org/>.

[FIPS-202]
National Institute of Standards and Technology, "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", 2015, <https://doi.org/10.6028/NIST.FIPS.202>.

[McEliece]
McEliece, R., "A Public-key Cryptosystem based on Algebraic Coding Theory", DSN Progress Report 42-44, 1978.

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

[RFC8019] Nir, Y. and V. Smyslov, "Protecting Internet Key Exchange Protocol Version 2 (IKEv2) Implementations from Distributed Denial-of-Service Attacks", RFC 8019, DOI 10.17487/RFC8019, November 2016, <https://www.rfc-editor.org/info/rfc8019>.

[RFC8229] Pauly, T., Touati, S., and R. Mantha, "TCP Encapsulation of IKE and IPsec Packets", RFC 8229, DOI 10.17487/RFC8229, August 2017, <https://www.rfc-editor.org/info/rfc8229>.

Authors' Addresses

CJ Tjhai
Post-Quantum
UK

Email: cjt@post-quantum.com

Tobias Heider
genua GmbH
DE

Email: me@tobhe.de

Valery Smyslov
ELVIS-PLUS
PO Box 81
Moscow (Zelenograd) 124460
RU

Phone: +7 495 276 0211
Email: svan@elvis.ru