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.

Table of Contents

1. Introduction 2
1.1. Requirements Language 3
2. Performance bottlenecks 3
3. Negotiation of performance specific Child SAs 3
4. Implementation specifics 4
4.1. One Child per CPU 4
4.2. QoS Child SA's 5
5. Payload Format 6
5.1. NUM_QUEUES Notify Payload 6
5.2. QUEUE_INFO Notify Payload 6
6. Security Considerations 7
7. Implementation Status 7
7.1. Linux XFRM 8
7.2. Libreswan 8
7.3. strongSWAN 9
8. IANA Considerations 9
9. References 9
9.1. Normative References 9
9.2. Informative References 10
Authors' Addresses 10

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

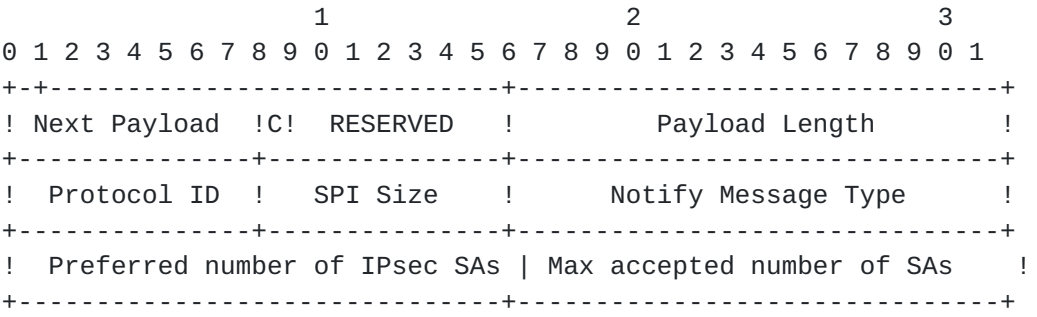
being rekeyed. [This kind of suggests this should be a TS_TYPE and not a Notify]

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.



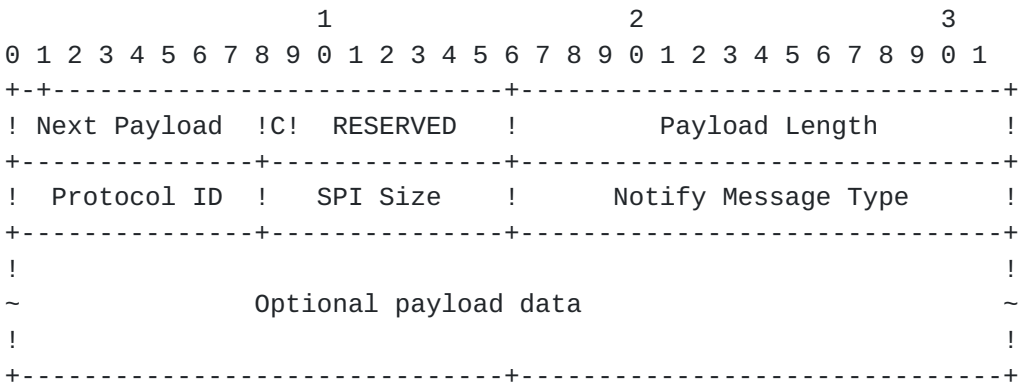
- Protocol ID (1 octet) - MUST be 0. MUST be ignored if not 0.
- SPI Size (1 octet) - MUST be 0. MUST be ignored if not 0. by the IPsec protocol ID
- Notify Message Type (2 octets) - set to [TBD]
- Preferred number of per-CPU IPsec SAs (2 octets). Value MUST be greater than 0. If 0 is received, it MUST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