Network   A. Antony
Internet-Draft   S. Klassert
Intended status: Standards Track   secunet
Expires: August 26, 2021   P. Wouters
                                                                 Red Hat
                                                       February 22, 2021

IKEv2 support for per-queue Child SAs
draft-pwouters-multi-sa-performance-01

Abstract

This document defines two Notification Payloads for the Internet Key Exchange Protocol Version 2 (IKEv2): NUM_QUEUES and QUEUE_INFO. These payloads add support for indicating that the negotiating of multiple identical Child SAs are to be used to optimize performance based on the number of queues or CPUs, or to create multiple Child SAs for different Quality of Service (QoS) levels. It indicates that a newer idetnical Child SA should not be interpreted as a replacement Child SA.

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

Status of This Memo

Copyright Notice

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 because the QoS type is not a part of the TS. The result is that IPsec can't do active Quality of Service prioritizing without disabling the anti replay detection.

While this could be mitigated by setting up multiple narrowed Child SA's, for example using Populate From Packet (PFP) as specified in [RFC4301], this IPsec feature is not widely implemented.

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 is useful to be able to convey how many Child SAs are required for optimized traffic. This avoids triggering CREATE_CHILD_SA exchanges that will only be rejected by the peer.

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 practical 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 beyond the initial Child SA. Both peers send their minimum number of Child SAs they prefer to install. Both peers pick the maximum iof the two numbers (within reason). That is if one peer prefers 16 and the other peer prefers 48, then the number negotiated is 48. If a 49th Child SA is

attempted with QUEUE_INFO notify payload, it can be rejected using TS_UNACCEPTABLE.

The NUM_QUEUES Notify payload is sent as part of the IKE_AUTH or as part of an CREATE_CHILD_SA Exchange for an initial new Child SA request. It identifies the initial Child SA of a set, and allows the peers to ensure that the initial Child SA (or its rekeyed version) remains active for the lifetime of the IPsec connection. Further CREATE_CHILD_SA messages for subsequent copies of the original Child SA MUST NOT contain the NUM_QUEUES notify payload. This initial Child SA (or its REKEYed successor) MUST remain active for the lifetime of the IPsec session to ensure there is always a CHILD SA that can be selected to send traffic over. Subsequent Child SA's can be installed with an additional selector, such as CPU or queue, or ToS value.

The QUEUE_INFO Notify MUST be sent in CREATE_CHILD_SA for subsequent copies of the original Child SA. It is used to indicate the queue or CPU or QoS value of this specific copy of the initial Child SA. These additional Child SA's can be started on-demand or all at once and can also be deleted if a peer deems this specific queue or CPU or QoS value to be idle. During CREATE_CHILD_SA's sent for Child SA rekey, the QUEUE_INFO Notify MUST NOT be included. As with Traffic Selector payloads, the QUEUE_INFO may not be different from the Child SA being rekeyed.

This implies a CREATE_CHILD_SA exchange can only have either a QUEUE_INFO or NUM_QUEUES notify. If both Notify types are received, NUM_QUEUES has precedence and QUEUE_INFO MUST be ignored.

The NUM_QUEUES notify, even though it can be sent in IKE_AUTH exchange with TS, is not an attribute of the IKE peer. It is an attribute of the Child SA, similar as how the USE_TRANSPORT notify payload. This allows an IKE peer to have multiple Child SA's covering different traffic selectors and selectively decide whether or not to use multiple Child SA's for those different Child SA's.

4. Implementation specifics

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

4.1. One CPU per Child

A simple distribution could be to install one Child SA on each 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 role is

performed by the initial Child SA of the set of identical Child SAs. 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 installed in such a way to only be used 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 ACQUIRE message does not support sending a per-CPU identifier, then the IKE daemon may initiate all its Child SAs immediately upon receiving an ACQUIRE.

Performing per-CPU Child SA negotiations can result in both peers initiating additional Child SAs at once. This is especially likely in the per-CPU acquire case. Responders should install the additional Child SA on a CPU with the least amount of additional Child SA's for this TSi/TSr pair. It should count outstanding ACQUIREs as an assigned additional Child 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 Child SA, MAY send a delete to delete the duplicate additional Child SA.

As an optimization, additional Child SAs that see little traffic MAY be deleted. The initial Child SA that is not limited to a single CPU MUST NOT be deleted when idle, as it is likely to be idle if enough per-CPU Child SA's are installed. However, if one of those per-CPU child SA's is deleted because it was idle, and subsequently that CPU starts the generate traffic again, that traffic should be encrypted by the initial non-CPU specific Child SA while the IKE daemon processes the ACQUIRE to bring up a new per-CPU Child SA.

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 that Child SA to send traffic. However, it MUST install all inbound Child SA's as it has commited to receiving traffic on these negotiated Child SAs. It MUST NOT generate an error when deleting the (missing) outbound SA component of such a Child SA.

A per-CPU ACQUIRE message SHOULD still send the Traffic Selector (TSi) entry containing the information of the trigger packet . This information MAY be used by the peer to select the most optimal target CPU to install the additional Child SA on. For example, if the trigger packet was for a TCP destination to port 25 (SMTP), it might be able to install the Child SA on the CPU that is also running the

mail server process. Trigger packet Traffic Selectors are documented in [RFC7296] Section 2.9.

The QUEUE_INFO Notify payload MUST be sent in the CREATE_CHILD_SA request for the additional Child SAs. It is used to convey the QoS stream or CPU id. Note that this ID value does not neccessarilly have to match any physical CPU IDs.

[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 may use multiple source ports. We need define a way to select UDP source ports for the Sub SA while IKE SA and the Head remain on UDP port 4500 - 4500. NOTE: libreswan has an expirmental implementation for Linux XFRM.]

[Add text about how this parallel SA use may inter operate with 6311? may be not?]

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 responder, TS_UNACCEPTABLE MUST be returned. During Child SA REKEY, the QUEUE_INFO Notify MUST NOT be included and MUST be ignored when received.

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

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-----------------------------+-------------------------------+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+---------------+---------------+-------------------------------+
!  Protocol ID  !   SPI Size    !      Notify Message Type      !
+---------------+---------------+-------------------------------+
!  Minimum number of IPsec SAs                                  !
+-------------------------------+-------------------------------+
```

o Protocol ID (1 octet) - MUST be 0. MUST be ignored if not 0.

o SPI Size (1 octet) - MUST be 0. MUST be ignored if not 0.

o Notify Message Type (2 octets) - set to [TBD]

o Minimum number of per-CPU IPsec SAs (4 octets). initiator value 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 is not counted as part of these numbers.

5.2. QUEUE_INFO Notify Payload

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-----------------------------+-------------------------------+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+---------------+---------------+-------------------------------+
!  Protocol ID  !   SPI Size    !      Notify Message Type      !
+---------------+---------------+-------------------------------+
!                                                               !
~                Optional payload data                          ~
!                                                               !
+-------------------------------+-------------------------------+
```

o Protocol ID (1 octet) - MUST be 0. MUST be ignored if not 0.

o SPI Size (1 octet) - MUST be 0. MUST be ignored if not 0.

o Notify Message Type (2 octets) - set to [TBD]

o Optional Payload Data. This can be set to identify the QoS value or the CPU ID. The interpretation of the value is left to local

implementations? [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: Implements Initial Child SA and per-CPU additional Child SA's. Also implements per-CPU ACQUIRES using NETLINK. PFKEYv2 is not supported.

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 Initial Child SA and per-CPU additional Child SA's

Licensing: GPLv2

Implementation experience: TBD

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

7.3. strongSWAN

Organization: Secunet

Name: StrongSWAN https://github.com/antonyantony/strongswan/

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

Level of maturity: Alpha

Coverage: implements Initial Child SA and per-CPU additional Child SA's

Licensing: GPLv2

Implementation experience: the Linux XFRM implemenation needs an addtional flag on the SPD entry, XFRM_POLICY_CPU_ACQUIRE. It should be set only on the "outgoing" policy. The flag should be disabled when the policy is a trap policy without SPD state.

After a successfull negotiation of NUM_QUEUES, the SPD policy is updated to enable the XFRM_POLICY_CPU_ACQUIRE flag. For the outgoing additional Child SAs, the u32 XFRMA_SA_PCPU attribute is set, starting from 0. The incoming SA do not need XFRMA_SA_PCPU. The kernel internally set the value 0xFFFFFF. The strongswan implentation uses private space values for NUM_QUEUES (40970) and QUEUE_INFO (40971). The iproute2 software that supporte these two attributes is available at https://github.com/antonyantony/iproute2/tree/pcpu-v1

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

8. IANA Considerations

This document defines two new IKEv2 Notify messages 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

[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>.

[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