Workgroup: Network
Internet-Draft:
draft-ietf-ipsecme-multi-sa-performance-06
Published: 19 March 2024
Intended Status: Standards Track
Expires: 20 September 2024
Authors: A. Antony   T. Brunner   S. Klassert   P. Wouters
secunet   codelabs   secunet   Aiven

IKEv2 support for per-resource Child SAs

Abstract

This document defines two Notify Message Type Payloads for the Internet Key Exchange Protocol Version 2 (IKEv2) to support the negotiation of multiple Child SAs with the same Traffic Selectors used on different resources, such as CPUs, to increase bandwidth of IPsec traffic between peers.

The SA_RESOURCE_INFO notification is used to convey information that the negotiated Child SA and subsequent new Child SAs with the same Traffic Selectors are a logical group of Child SAs where most or all of the Child SAs are bound to a specific resource, such as a specific CPU. The TS_MAX_QUEUE notify conveys that the peer is unwilling to create more additional Child SAs for this particular negotiated Traffic Selector combination.

Using multiple Child SAs with the same Traffic Selectors has the benefit that each resource holding the Child SA has its own Sequence Number Counter, ensuring that CPUs don't have to synchronize their cryptographic state or disable their packet replay protection.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Most IPsec implementations are currently limited to using one hardware queue or a single CPU resource for a Child SA. Running packet stream encryption in parallel can be done, but there is a bottleneck of different parts of the hardware locking or waiting to get their sequence number assigned for the packet it is encrypting. The result is that a machine with many such resources is limited to only using one of these resources per Child SA. This severely limits the throughput that can be attained. For example, at the time of writing, an unencrypted link of 10Gbps or more is commonly reduced to 2-5Gbps when IPsec is used to encrypt the link using AES-GCM. By using the implementation specified in this document, aggregate throughput increased from 5Gbps using 1 CPU to 40-60 Gbps using 25-30 CPUs.

While this could be (partially) mitigated by setting up multiple narrowed Child SAs, for example using Populate From Packet (PFP) as specified in IPsec Architecture [RFC4301], this IPsec feature would cause too many Child SAs (one per network flow) or too few Child SAs (one network flow used on multiple CPUs). PFP is also not widely implemented.

To make better use of multiple network queues and CPUs, it can be beneficial to negotiate and install multiple Child SAs with identical Traffic Selectors. IKEv2 [RFC7296] already allows installing multiple Child SAs with identical Traffic Selectors, but it offers no method to indicate that the additional Child SA is being requested for performance increase reasons and is restricted to some resource (queue or CPU).

When an IKEv2 peer is receiving additional Child SA's for a single set of Traffic Selectors than it is willing to create, it can return an error notify of TS_MAX_QUEUE.

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. Terminology

This document uses the following terms defined in IKEv2 [RFC7296]: Notification Data, Traffic Selectors (TS), TSi/TSr, Child SA, Configuration Payload (CP), IKE SA, CREATE_CHILD_SA and NO_ADDITIONAL_SAS.

2. Performance bottlenecks

There are a number of practical reasons why most implementations have to limit a Child SA to only one specific hardware resource, but a key limitation is that sharing the cryptographic state, counters and sequence numbers between multiple CPUs that are trying to use these shared states at the same time is not feasible without a significant performance penalty. There is a need to negotiate and establish multiple Child SAs with identical TSi/TSr on a per-resource basis.

3. Negotiation of CPU specific Child SAs

An initial IKEv2 exchange is used to setup an IKE SA and the initial Child SA. If multiple Child SAs with the same Traffic Selectors that are bound to a single resource are desired, the initiator will add the SA_RESOURCE_INFO notify payload to the Exchange negotiating the Child SA (eg IKE_AUTH or CREATE_CHILD_SA). If this initial Child SA will be tied to a specific resource, it MAY indicate this by including an identifier in the Notification Data. A responder that is willing to have multiple Child SAs for the same Traffic Selectors will respond by also adding the SA_RESOURCE_INFO notify payload in which it MAY add a non-zero Notify Data.

Additional resource-specific Child SAs are negotiated as regular Child SAs using the CREATE_CHILD_SA exchange and are similarly identified by an accompanying SA_RESOURCE_INFO notification.

Upon installation, each resource-specific Child SA is associated with an additional local selector, such as CPU or queue. These resource-specific Child SAs MUST be negotiated with identical Child SA properties that were negotiated for the initial Child SA. This includes cryptographic algorithms, Traffic Selectors, Mode (e.g. transport mode), compression usage, etc. However, each Child SA does have its own keying material that is individually derived according to the regular IKEv2 process. The SA_RESOURCE_INFO notify payload MAY be empty or MAY contain some identifying data. This identifying data SHOULD be a unique identifier within all the Child SAs with the same TS payloads and the peer MUST only use it for debugging purposes.

Additional Child SAs can be started on-demand or can be started all at once. Peers may also delete specific per-resource Child SAs if they deem the associated resource to be idle.

During the CREATE_CHILD_SA rekey for the Child SA, the SA_RESOURCE_INFO notification MAY be included, but regardless of whether or not it is included, the rekeyed Child SA should be bound to the same resource(s) as the Child SA that is being rekeyed.

4. Implementation Considerations

There are various considerations that an implementation can use to determine the best way to install multiple Child SAs.

A simple distribution could be to install one additional Child SA on each CPU. An implementation MAY ensure that one Child SA can be used by all CPUs, so that while negotiating a new per-CPU Child SA, which typically takes a 1RTT delay, the CPU with no CPU-specific Child SA can still encrypt its packets using the Child SA that is available for all CPUs. Alternatively, if an implementation finds it needs to encrypt a packet but the current CPU does not have the resources to encrypt this packet, it can relay that packet to a specific CPU that does have the capability to encrypt the packet, although this will come with a performance penalty.

Performing per-CPU Child SA negotiations can result in both peers initiating additional Child SAs at once. This is especially likely if per-CPU Child SAs are triggered by individual SADB_ACQUIRE [RFC2367] messages. Responders should install the additional Child SA on a CPU with the least amount of additional Child SAs for this TSi/TSr pair.

When the number of queue or CPU resources are different between the peers, the peer with the least amount of resources may decide to not install a second outbound Child SA for the same resource as it will never use it to send traffic. However, it MUST install all inbound Child SAs as it has committed to receiving traffic on these negotiated Child SAs.

If per-CPU packet trigger (eg SADB_ACQUIRE) messages are implemented (see Section 6), the Traffic Selector (TSi) entry containing the information of the trigger packet SHOULD be included in the TS set similarly to regular Child SAs as specified in IKEv2 [RFC7296] Section 2.9. Based on the trigger TSi entry, an implementations can 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 IKEv2 [RFC7296] Section 2.9.

As per IKEv2, rekeying a Child SA SHOULD use the same (or wider) Traffic Selectors to ensure that the new Child SA covers everything that the rekeyed Child SA covers. This includes Traffic Selectors negotiated via Configuration Payloads (CP) such as INTERNAL_IP4_ADDRESS which may use the original wide TS set or use the narrowed TS set.

5. Payload Format

The Notify Payload format is defined in IKEv2 [RFC7296] section 3.10, and is copied here for convenience.

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. SA_RESOURCE_INFO Notify Status Message 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      !
+---------------+---------------+-------------------------------+
!                                                               !
~                Resource Identifier (optional)                 ~
!                                                               !
+-------------------------------+-------------------------------+
```

*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.

*Notify Status Message Type (2 octets) - set to [TBD1].

*Resource Identifier (optional). This opaque data may be set to convey the local identity of the resource.

5.2. TS_MAX_QUEUE Notify Error Message 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      !
+---------------+---------------+-------------------------------+
```

*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.

*Notify Error Message Type (2 octets) - set to [TBD2]

*There is no data associated with this Notify type.

6. Operational Considerations

Implementations supporting per-CPU SAs SHOULD extend their local SPD selector, and the mechanism of on-demand negotiation that is triggered by traffic to include a CPU (or queue) identifier in their packet trigger (eg SADB_ACQUIRE) message from the SPD to the IKE daemon. An implementation which does not support receiving per-CPU packet trigger messages MAY initiate all its Child SAs immediately upon receiving the (only) packet trigger message it will receive from the IPsec stack. Such implementations also need to be careful when receiving a Delete Notify request for a per-CPU Child SA, as it has no method to detect when it should bring up such a per-CPU Child SA again later. And bringing the deleted per-CPU Child SA up again immediately after receiving the Delete Notify might cause an infinite loop between the peers. Another issue of not bringing up all its per-CPU Child SAs is that if the peer acts similarly, the two peers might end up with only the first Child SA without ever activating any per-CPU Child SAs. It is there for RECOMMENDED to implement per-CPU packet trigger messages.

Peers SHOULD be flexible with the maximum number of Child SAs they allow for a given TSi/TSr combination to account for corner cases. For example, during Child SA rekeying, there might be a large number of additional Child SAs created before the old Child SAs are torn down. Similarly, when using on-demand Child SAs, both ends could trigger multiple Child SA requests as the initial packet causing the Child SA negotiation might have been transported to the peer via the first Child SA where its reply packet might also trigger an on-demand Child SA negotiation to start. As additional Child SAs consume little additional resources, allowing at the very least double the number of available CPUs is RECOMMENDED. An implementation MAY allow unlimited additional Child SAs and only limit this number based on its generic resource protection strategies that are used to require COOKIES or refuse new IKE or Child SA negotiations. Although having a very large number (eg hundreds or thousands) of SAs may slow down per-packet SAD lookup.

Implementations might support dynamically moving a per-CPU Child SAs from one CPU to another CPU. If this method is supported, implementations must be careful to move both the inbound and outbound SAs. If the IPsec endpoint is a gateway, it can move the inbound SA and outbound SA independently from each other. It is likely that for a gateway, IPsec traffic would be asymmetric. If the IPsec endpoint is the same host responsible for generating the traffic, the inbound and outbound SAs SHOULD remain as a pair on the same CPU. If a host previously skipped installing an outbound SA because it would be an unused duplicate outbound SA, it will have to create and add the previously skipped outbound SA to the SAD with the new CPU ID. The inbound SA may not have CPU ID in the SAD. Adding the outbound SA to the SAD requires access to the key material, whereas for updating the CPU selector on an existing outbound SAs access to key material might not be needed. To support this, the IKE software might have to hold on to the key material longer than it normally would, as it might actively attempt to destroy key material from memory that the IKE daemon no longer needs access to.

An implementation that does not accept any further resource specific Child SAs MUST NOT return the NO_ADDITIONAL_SAS error because this can be interpreted by the peer that no other Child SAs with different TSi/TSr are allowed either. Instead, it MUST return TS_MAX_QUEUE.

7. Security Considerations

Similar to how an implementation should limit the number of half-open SAs to limit the impact of a denial of service attack, it is RECOMMENDED that an implementation limits the maximum number of additional Child SAs allowed per unique TSi/TSr.

Using multiple resource specific child SAs makes sense for high volume IPsec connections on IPsec gateway machines where the administrator has a trust relationship with the peer's administrator and abuse is unlikely and easily escalated to resolve.

This trust relationship is usually not present for the Remote Access VPN type deployments, and allowing per-CPU Child SA's is NOT RECOMMENDED in these scenarios. Therefore, it is also NOT RECOMMENDED to allow per-CPU Child SAs per default.

The SA_RESOURCE_INFO notify contains an optional data payload that can be used by the peer to identify the Child SA belonging to a specific resource. The notify data SHOULD NOT be an identifier that can be used to gain information about the hardware. For example, using the CPU number itself as identifier might give an attacker knowledge which packets are handled by which CPU ID and it might optimize a brute force attack against the system.

8. 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].

8.1. Linux XFRM

Organization: Linux kernel XFRM

Name: XFRM-PCPU-v3 https://git.kernel.org/pub/scm/linux/kernel/git/klassert/linux-stk.git/log/?h=xfrm-pcpu-v3

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

Level of maturity: Alpha

Coverage: Implements a general Child SA and per-CPU Child SAs. It only supports the NETLINK API. The PFKEYv2 API is not supported.

Licensing: GPLv2

Implementation experience: The Linux XFRM implementation added two additional attributes to support per-CPU SAs. There is a new attribute XFRMA_SA_PCPU, u32, for the SAD entry. This attribute should present on the outgoing SA, per-CPU Child SAs, starting from 0. This attribute MUST NOT be present on the first XFRM SA. It is used by the kernel only for the outgoing traffic, (clear to encrypted). The incoming SAs do not need XFRMA_SA_PCPU attribute. XFRM stack can not use CPU id on the incoming SA. The kernel internally sets the value to 0xFFFFFF for the incoming SA and the initial Child SA that can be used by any CPU. However, one may add XFRMA_SA_PCPU to the incoming per-CPU SA to steer the ESP flow, to a specific Q or CPU e.g ethtool ntuple configuration. The SPD entry has new flag XFRM_POLICY_CPU_ACQUIRE. It should be set only on the "out" policy. The flag should be disabled when the policy is a trap policy, without SPD entries. After a successful negotiation of CPU_QUEUES, while adding the first Child SA, the SPD entry can be updated with the XFRM_POLICY_CPU_ACQUIRE flag. When XFRM_POLICY_CPU_ACQUIRE is set, the XFRM_MSG_ACQUIRE generated will include the XFRMA_SA_PCPU attribute.

Contact: Steffen Klassert steffen.klassert@secunet.com

8.2. 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 combining a regular (all-CPUs) Child SA and per-CPU additional Child SAs

Licensing: GPLv2

Implementation experience: TBD

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

8.3. strongSwan

Organization: The StrongSwan Project

Name: StrongSwan https://github.com/strongswan/strongswan/tree/per-cpu-sas-poc/

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

Level of maturity: Alpha

Coverage: implements combining a regular (all-CPUs) Child SA and per-CPU additional Child SAs

Licensing: GPLv2

Implementation experience: StrongSwan use private space values for notifications CPU_QUEUES (40970) and QUEUE_INFO (40971).

Contact: Tobias Brunner tobias@strongswan.org

8.4. iproute2

Organization: The iproute2 Project

Name: iproute2 https://github.com/antonyantony/iproute2/tree/pcpu-v1

Description: Implemented the per-CPU attributes for the "ip xfrm" command.

Level of maturity: Alpha

Licensing:
GPLv2

Implementation experience: TBD

Contact: Antony Antony antony.antony@secunet.com

9. IANA Considerations

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

Value	Notify Messages - Status Types	Reference
[TBD1]	SA_RESOURCE_INFO	[this document]

Figure 1

This document defines one new registration for the IANA "IKEv2 Notify Messages Types - Error Types" registry.

Value	Notify Messages - Error Types	Reference
[TBD2]	TS_MAX_QUEUE	[this document]

Figure 2

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>.

[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>.

10.2. Informative References

[RFC2367] McDonald, D., Metz, C., and B. Phan, "PF_KEY Key Management API, Version 2", RFC 2367, DOI 10.17487/

RFC2367, July 1998, <https://www.rfc-editor.org/info/rfc2367>.

[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

Tobias Brunner
codelabs GmbH

Email: tobias@codelabs.ch

Steffen Klassert
secunet Security Networks AG

Email: steffen.klassert@secunet.com

Paul Wouters
Aiven

Email: paul.wouters@aiven.io