Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: October 21, 2017

S. Fluhrer
D. McGrew
P. Kampanakis
Cisco Systems
April 19, 2017

Postquantum Preshared Keys for IKEv2
draft-fluhrer-qr-ikev2-04

Abstract

The possibility of quantum computers pose a serious challenge to cryptography algorithms widely today. IKEv2 is one example of a cryptosystem that could be broken; someone storing VPN communications today could decrypt them at a later time when a quantum computer is available. It is anticipated that IKEv2 will be extended to support quantum secure key exchange algorithms; however that is not likely to happen in the near term. To address this problem before then, this document describes an extension of IKEv2 to allow it to be resistant to a Quantum Computer, by using preshared keys.

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 http://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 October 21, 2017.

Copyright Notice

Copyright (c) 2017 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 (http://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

It is an open question whether or not it is feasible to build a quantum computer (and if so, when might one be implemented), but if it is, many of the cryptographic algorithms and protocols currently in use would be insecure. A quantum computer would be able to solve DH and ECDH problems, and this would imply that the security of existing IKEv2 systems would be compromised. IKEv1 when used with strong preshared keys is not vulnerable to quantum attacks, because those keys are one of the inputs to the key derivation function. If the preshared key has sufficient entropy and the PRF, encryption and authentication transforms are postquantum secure, then the resulting system is believed to be quantum resistant, that is, believed to be invulnerable to an attacker with a Quantum Computer.

This document describes a way to extend IKEv2 to have a similar property; assuming that the two end systems share a long secret key, then the resulting exchange is quantum resistant. By bringing postquantum security to IKEv2, this note removes the need to use an obsolete version of the Internet Key Exchange in order to achieve that security goal.

The general idea is that we add an additional secret that is shared between the initiator and the responder; this secret is in addition

to the authentication method that is already provided within IKEv2. We stir in this secret into the SK_d value, which is used to generate the key material (KEYMAT) keys and the SKEYSEED for the child SAs; this secret provides quantum resistance to the IPsec SAs (and any child IKE SAs). We also stir in the secret into the SK_pi, SK_pr values; this allows both sides to detect a secret mismatch cleanly.

It was considered important to minimize the changes to IKEv2. The existing mechanisms to do authentication and key exchange remain in place (that is, we continue to do (EC)DH, and potentially a PKI authentication if configured). This does not replace the authentication checks that the protocol does; instead, it is done as a parallel check.

1.1. Changes

Changes in this draft from the previous versions

draft-03

- Modified how we stir the PPK into the IKEv2 secret state

- Modified how the use of PPKs is negotiated

draft-02

- Simplified the protocol by stirring in the preshared key into the child SAs; this avoids the problem of having the responder decide which preshared key to use (as it knows the initiator identity at that point); it does mean that someone with a Quantum Computer can recover the initial IKE negotation.

- Removed positive endorsements of various algorithms. Retained warnings about algorithms known to be weak against a Quantum Computer

draft-01

- Added explicit guidance as to what IKE and IPsec algorithms are Quantum Resistant

draft-00

- We switched from using vendor ID's to transmit the additional data to notifications

- We added a mandatory cookie exchange to allow the server to communicate to the client before the initial exchange

- We added algorithm agility by having the server tell the client what algorithm to use in the cookie exchange

- We have the server specify the PPK Indicator Input, which allows the server to make a trade-off between the efficiency for the search of the clients PPK, and the anonymity of the client.

- We now use the negotiated PRF (rather than a fixed HMAC-SHA256) to transform the nonces during the KDF

1.2. Requirements Language

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].

2. Assumptions

We assume that each IKE peer has a list of Postquantum Preshared Keys (PPK) along with their identifiers (PPK_id), and any potential IKE initiator has a selection of which PPK to use with with any specific responder. In addition, the implementation has a configurable flag that determines whether this postquantum preshared key is mandatory. This PPK is independent of the preshared key (if any) that the IKEv2 protocol uses to perform authentication.

3. Exchanges

If the initiator is configured to use a postquantum preshared key with the responder (whether or not the use of the PPK is optional), then it will include a notify payload in the initial exchange as follows:

```
Initiator                          Responder
-----------------------------------------------------------------
HDR, SAi1, KEi, Ni, N(PPK_SUPPORT)  --->
```

N(PPK_SUPPORT) is a status notification payload with the type [TBA]; it has a protocol ID of 0, and no SPI and no notification data associated with it.

If the initiator needs to resend this initial message with a cookie (because the responder response included a cookie notification), then the resend would include the PPK_SUPPORT notification if the original message did.

When the responder receives this initial exchange with the notify, then it MUST check if has a PPK configured. If it does, it MUST

reply with the IKE initial exchange including a notification in response.

```
Initiator                          Responder
-------------------------------------------------------------------
                <--- HDR, SAr1, KEr, Nr, [CERTREQ], N(PPK_SUPPORT)
```

If the responder does not have a PPK configured, then it continues with the IKE protocol as normal, not including the notify.

When the initiator receives this reply, it checks whether the responder included the PPK_SUPPORT notify. If the responder did not, then the initiator MUST either proceed with the standard IKE negotiation (without using a PPK), or abort the exchange (for example, because the initiator has the PPK marked as mandatory). If the responder did include the PPK_SUPPORT notify, then it selects a PPK, along with its identifier PPK_id. Then, it computes this modification of the standard IKE key derivation:

```
 SKEYSEED = prf(Ni | Nr, g^ir)
 {SK_d' | SK_ai | SK_ar | SK_ei | SK_er | SK_pi' | SK_pr' )
                  = prf+ (SKEYSEED, Ni | Nr | SPIi | SPIr }
 SK_d = prf(PPK, SK_d')
 SK_pi = prf(PPK, SK_pi')
 SK_pr = prf(PPK, SK_pr')
```

That is, we use the standard IKE key derivation process except that the three subkeys SK_d, SK_pi, SK_pr are run through the prf again, this time using the PPK as the key.

The initiator then sends the initial encrypted message, including the PPK_id value as follows:

```
Initiator                          Responder
-------------------------------------------------------------------
HDR, SK {IDi, [CERT,] [CERTREQ,]
    [IDr,] AUTH, SAi2,
    TSi, TSr, N(PPK_IDENTITY)(PPK_id)}  --->
```

N(PPK_IDENITY) is a status notification payload with the type [TBA]; it has a protocol ID of 0, and no SPI and has a notification data that consists of the identifier PPK_id.

When the responder receives this encrypted exchange, it first computes the values:

```
SKEYSEED = prf(Ni | Nr, g^ir)
{SK_d' | SK_ai | SK_ar | SK_ei | SK_er | SK_pi' | SK_pr' }
                = prf+ (SKEYSEED, Ni | Nr | SPIi | SPIr )
```

It then uses the SK_ei value to decrypt the message; and then finds the PPK_id value attached to the notify. It then scans through the payload for the PPK_id attached to the N(PPK_IDENTITY); if it has no such PPK, it fails the negotiation. If it does have a PPK with that identity, it further computes:

```
SK_d = prf(PPK, SK_d')
SK_pi = prf(PPK, SK_pi')
SK_pr = prf(PPK, SK_pr')
```

And computes the enchange (validating the AUTH payload that the initiator included) as standard.

This table summarizes the above logic by the responder

Received PPK_SUPPORT	Have PPK	PPK Mandatory	Action
No	No	*	Standard IKE protocol
No	Yes	No	Standard IKE protocol
No	Yes	Yes	Abort negotiation
Yes	No	*	Standard IKE protocol
Yes	Yes	*	Include PPK_SUPPORT

When the initiator receives the response, then (if it is configured to use a PPK with the responder), then it checks for the presense of the notification. If it receives one, it marks the SA as using the configured PPK to generate SK_d, SK_pi, SK_pr (as shown above); if it does not receive one, it MUST either abort the exchange (if the PPK was configured as mandatory), or it MUST continue without using the PPK (if the PPK was configured as optional).

If the initial exchange had PPK_SUPPORT sent by both the initiator and the responder, and the initiator does not include a PPK_NOTIFY notification, then the responder SHOULD fail the exchange.

With this protocol, the computed SK_d is a function of the PPK, and assuming that the PPK has sufficient entropy (for example, at least 2**256 possible values), then even if an attacker were able to recover the rest of the inputs to the prf function, it would be infeasible to use Grover's algorithm with a Quantum Computer to recover the SK_d value. Similarly, every child SA key is a function of SK_d, hence all the keys for all the child SAs are also quantum resistant (assuming that the PPK was high entropy and secret, and that all the subkeys are sufficiently long). However, this quantum

resistance does not extend to the initial SK_ei, SK_er keys; an implementation MAY rekey the initial IKE SA immediately after negotiating it; this would reduce the amount of data available to an attacker with a Quantum Computer.

4. PPK ID format

This standard requires that both the initiator and the responder have a secret PPK value, with the responder selecting the PPK based on the PPK_ID that the initiator sends. In this initial standard, both the initator and the responder are configured with fixed PPK and PPK_ID values, and do the look up based on that. It is anticipated that later standards will extend this technique to allow dynamically changing PPK values. To facilitate such an extension, we specify that the PPK_ID that the initiator sends will have its first octet be the PPK ID Type value, which is encoded as follows:

PPK ID Type	Value
PPK_ID_OPAQUE	0
PPK_ID_FIXED	1
RESERVED TO IANA	2-127
Reserved for private use	128-255

For PPK_ID_OPAQUE, the format of the PPK ID (and the PPK itself) is not specified by this document; it is assumed to be mutually intelligible by both by initiator and the responder. This PPK ID type is intended for those implementations that choose not to disclose the type of PPK to active attackers.

For PPK_ID_FIXED, the format of the PPK ID and the PPK are fixed octet strings; the remaining bytes of the PPK_ID are a configured value. We assume that there is a fixed mapping between PPK_ID and PPK, which is configured locally to both the initiator and the responder. The responder can use to do a look up the passed PPK_id value to determine the corresponding PPK value. Not all implementations are able to configure arbitrary octet strings; to improve the potential interoperability, it is recommended that, in the PPK_ID_FIXED case, both the PPK and the PPK_ID strings be limited to the base64 character set, namely the 64 characters 0-9, A-Z, a-z, + and /.

The PPK ID type values 2-127 are reserved for IANA; values 128-255 are for private use among mutually consenting parties.

5. PPK Distribution

PPK_id's of the type PPK_ID_FIXED (and the corresponding PPKs) are assumed to be configured within the IKE device in an out-of-band fashion. While the method of distribution is a local matter, one suggestion would be to reuse the format within [RFC6030], with the Key Id field being the PPK_ID (without the 0x01 prefix for a PPK_ID_FIXED), and with the PPK being the secret, and the algorithm as PIN ("Algorithm=urn:ietf:params:xml:ns:keyprov:pskc:pin").

6. Upgrade procedure

This algorithm was designed so that someone can introduce PPKs into an existing IKE network without causing network disruption.

In the initial phase of the network upgrade, the network administrator would visit each IKE node, and configure:

- The set of PPKs (and corresponding PPK_id's) that this node would need to know

- For each peer that this node would initiate to, which PPK that we would use

- That the use of PPK is currently optional

With this configuration, the node will continue to operate with nodes that have not yet been upgraded. This is due to the PPK_SUPPORT notify; if the initiator has not been upgraded, it will not send the PPK_SUPPORT notify (and so the responder will know that we will not use a PPK); if the responder has not been upgraded, it will not send the PPK_SUPPORT notify (and so the initiator will know not to use a PPK). And, if both peers have been upgraded, they will both realize it, and in that case, the link will be quantum secure

As an optional second step, after all nodes have been upgraded, then the administrator may then go back through the nodes, and mark the use of PPK as mandatory. This will not affect the strength against a passive attacker; it would mean that an attacker with a Quantum Computer (which is sufficiently fast to be able to break the (EC)DH in real time would not be able to perform a downgrade attack).

7. Security Considerations

Quantum computers are able to perform Grover's algorithm; that effectively halves the size of a symmetric key. Because of this, the user SHOULD ensure that the postquantum preshared key used has at

least 256 bits of entropy, in order to provide a 128 bit security level.

Although this protocol preserves all the security properties of IKE against adversaries with conventional computers, this protocol allows an adversary with a Quantum Computer to decrypt all traffic encrypted with the initial IKE SA. In particular, it allows the adversary to recover the identities of both sides. If there is IKE traffic other than the identities that need to be protected against such an adversary, one suggestion would be to form an initial IKE SA (which is used to exchange identities), perhaps by using the protocol documented in RFC6023. Then, you would immediately create a child IKE SA (which is used to exchange everything else). Because the child IKE SA keys are a function of SK_d, which is a function of the PPK (among other things), traffic protected by that SA is secure against Quantum capable adversaries.

In addition, the policy SHOULD be set to negotiate only quantum-resistant symmetric algorithms; while this RFC doesn't claim to give advise as to what algorithms are secure (as that may change based on future cryptographical results), here is a list of defined IKEv2 and IPsec algorithms that should NOT be used, as they are known not to be Quantum Resistant

Any IKE Encryption algorithm, PRF or Integrity algorithm with key size <256 bits

Any ESP Transform with key size <256 bits

PRF_AES128_XCBC and PRF_AES128_CBC; even though they are defined to be able to use an arbitrary key size, they convert it into a 128 bit key internally

8. References

8.1. Normative References

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <http://www.rfc-editor.org/info/rfc2104>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://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, <http://www.rfc-editor.org/info/rfc7296>.

8.2. Informational References

[RFC6023] Nir, Y., Tschofenig, H., Deng, H., and R. Singh, "A Childless Initiation of the Internet Key Exchange Version 2 (IKEv2) Security Association (SA)", RFC 6023, DOI 10.17487/RFC6023, October 2010, <http://www.rfc-editor.org/info/rfc6023>.

[RFC6030] Hoyer, P., Pei, M., and S. Machani, "Portable Symmetric Key Container (PSKC)", RFC 6030, DOI 10.17487/RFC6030, October 2010, <http://www.rfc-editor.org/info/rfc6030>.

[SPDP] McGrew, D., "A Secure Peer Discovery Protocol (SPDP)", 2001, <http://www.mindspring.com/~dmcgrew/spdp.txt>.

Appendix A. Discussion and Rationale

The idea behind this is that while a Quantum Computer can easily reconstruct the shared secret of an (EC)DH exchange, they cannot as easily recover a secret from a symmetric exchange this makes the SK_d, and hence the IPsec KEYMAT and any child SA's SKEYSEED, depend on both the symmetric PPK, and also the Diffie-Hellman exchange. If we assume that the attacker knows everything except the PPK during the key exchange, and there are 2**n plausible PPK's, then a Quantum Computer (using Grover's algorithm) would take O(2**(n/2)) time to recover the PPK. So, even if the (EC)DH can be trivially solved, the attacker still can't recover any key material (except for the SK_ei, SK_er, SK_ai, SK_ar values for the initial IKE exchange) unless they can find the PPK, and that's too difficult if the PPK has enough entropy (for example, 256 bits). Note that we do allow an attacker with a Quantum Computer to rederive the keying material for the initial IKE SA; this was a compromise to allow the responder to select the correct PPK quickly.

Another goal of this protocol is to minimize the number of changes within the IKEv2 protocol, and in particular, within the cryptography of IKEv2. By limiting our changes to notifications, and translating the nonces, it is hoped that this would be implementable, even on systems that perform much of the IKEv2 processing is in hardware.

A third goal was to be friendly to incremental deployment in operational networks, for which we might not want to have a global shared key, and also if we're rolling this out incrementally. This

is why we specifically try to allow the PPK to be dependent on the peer, and why we allow the PPK to be configured as optional.

A fourth goal was to avoid violating any of the security goals of IKEv2.

Appendix B. Acknowledgement

We would like to thank Tero Kivine, Valery Smyslov, Paul Wouters and the rest of the ipsecme working group for their feedback and suggestions for the scheme

Authors' Addresses

Scott Fluhrer
Cisco Systems

Email: sfluhrer@cisco.com

David McGrew
Cisco Systems

Email: mcgrew@cisco.com

Panos Kampanakis
Cisco Systems

Email: pkampana@cisco.com

IPSecME Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 29, 2018

Y. Nir
Dell EMC

October 26, 2017

Using Edwards-curve Digital Signature Algorithm (EdDSA) in the Internet Key Exchange (IKEv2)
draft-ietf-ipsecme-eddsa-04

Abstract

This document describes the use of the Edwards-curve digital signature algorithm in the IKEv2 protocol.

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 April 29, 2018.

Copyright Notice

Copyright (c) 2017 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 [RFC7296] can use arbitrary signature algorithms as described in [RFC7427]. The latter RFC defines the SIGNATURE_HASH_ALGORITHMS notification where each side of the IKE negotiation lists its supported hash algorithms. This assumes that all signature schemes involve a hashing phase followed by a signature phase. This made sense because most signature algorithms either cannot sign messages bigger than their key or truncate messages bigger than their key.

EdDSA ([RFC8032]) defines signature methods that do not require pre-hashing of the message. Unlike other methods, these accept arbitrary-sized messages, so no pre-hashing is required. These methods are called Ed25519 and Ed448, which respectively use the Edwards 25519 and the Edwards 448 ("Goldilocks") curves. Although that document also defines pre-hashed versions of these algorithm, those versions are not recommended for protocols where the entire to-be-signed message is available at once. See section 8.5 or RFC 8032 for that recommendation.

EdDSA defines the binary format of the signatures that should be used in the "Signature Value" field of the Authentication Data Format in section 3. The CURDLE PKIX document ([I.D-curdle-pkix]) defines the object identifiers (OIDs) for these signature methods. For convenience, these OIDs are repeated in Appendix A.

In order to signal within IKE that no hashing needs to be done, we define a new value in the SIGNATURE_HASH_ALGORITHMS notification, one that indicates that no hashing is performed.

1.1. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. The "Identity" Hash Identifier

This document defines a new value called "Identity" (value is 5) in the hash algorithm registry for use in the SIGNATURE_HASH_ALGORITHMS notification. Inserting this new value into the notification indicates that the receiver supports at least one signature algorithm that accepts arbitrary-sized messages such as Ed25519 and Ed448.

Ed25519 and Ed448 are only defined with the Identity hash, and MUST NOT be sent to a receiver that has not indicated support for the "Identity" hash.

The pre-hashed versions of Ed25519 and Ed448 (Ed25519ph and Ed448ph respectively) MUST NOT be used in IKE.

3. Security Considerations

The new "Identity" value is needed only for signature algorithms that accept an arbitrary-sized input. It MUST NOT be used if none of the supported and configured algorithms have this property. On the other hand there is no good reason to pre-hash the inputs where the signature algorithm has that property. For this reason implementations MUST have the "Identity" value in the SIGNATURE_HASH_ALGORITHMS notification when EdDSA is supported and configured. Implementations SHOULD NOT have other hash algorithms in the notification if all supported and configured signature algorithms have this property.

4. IANA Considerations

IANA has assigned the value 5 for the algorithm with the name "Identity" in the "IKEv2 Hash Algorithms" registry with this draft as reference.

Upon publication of this document IANA is requested to update the entry with this document as reference.

5. 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>.

[RFC7427] Kivinen, T. and J. Snyder, "Signature Authentication in the Internet Key Exchange Version 2 (IKEv2)", RFC 7427, DOI 10.17487/RFC7427, January 2015, <https://www.rfc-editor.org/info/rfc7427>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[I.D-curdle-pkix] Josefsson, S. and J. Schaad, "Algorithm Identifiers for Ed25519, Ed25519ph, Ed448, Ed448ph, X25519 and X448 for use in the Internet X.509 Public Key Infrastructure", September 2017, <https://tools.ietf.org/html/draft-ietf-curdle-pkix-06>.

Appendix A. ASN.1 Objects

The normative reference for the ASN.1 objects for Ed25519 and Ed448 is in [I.D-curdle-pkix]. They are repeated below for convenience.

A.1. ASN.1 Object for Ed25519

```
id-Ed25519 OBJECT IDENTIFIER ::= { 1.3.101.112 }
```

Parameters are absent. Length is 7 bytes.

Binary encoding: 3005 0603 2B65 70

A.2. ASN.1 Object for Ed448

```
id-Ed448 OBJECT IDENTIFIER ::= { 1.3.101.113 }
```

Parameters are absent. Length is 7 bytes.

Binary encoding: 3005 0603 2B65 71

Author's Address

Yoav Nir
Dell EMC
9 Andrei Sakharov St
Haifa 3190500
Israel

EMail: ynir.ietf@gmail.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019

S. Fluhrer
D. McGrew
P. Kampanakis
Cisco Systems
V. Smyslov
ELVIS-PLUS

July 2, 2018

Postquantum Preshared Keys for IKEv2
draft-ietf-ipsecme-qr-ikev2-04

Abstract

The possibility of Quantum Computers pose a serious challenge to cryptography algorithms deployed widely today. IKEv2 is one example of a cryptosystem that could be broken; someone storing VPN communications today could decrypt them at a later time when a Quantum Computer is available. It is anticipated that IKEv2 will be extended to support quantum secure key exchange algorithms; however that is not likely to happen in the near term. To address this problem before then, this document describes an extension of IKEv2 to allow it to be resistant to a Quantum Computer, by using preshared keys.

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 http://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 3, 2019.

Copyright Notice

Copyright (c) 2018 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 (http://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

It is an open question whether or not it is feasible to build a Quantum Computer (and if so, when one might be implemented), but if it is, many of the cryptographic algorithms and protocols currently in use would be insecure. A Quantum Computer would be able to solve DH and ECDH problems in polynomial time [I-D.hoffman-c2pq], and this would imply that the security of existing IKEv2 [RFC7296] systems would be compromised. IKEv1 [RFC2409], when used with strong preshared keys, is not vulnerable to quantum attacks, because those keys are one of the inputs to the key derivation function. If the preshared key has sufficient entropy and the PRF, encryption and authentication transforms are postquantum secure, then the resulting system is believed to be quantum resistant, that is, invulnerable to an attacker with a Quantum Computer.

This document describes a way to extend IKEv2 to have a similar property; assuming that the two end systems share a long secret key, then the resulting exchange is quantum resistant. By bringing postquantum security to IKEv2, this note removes the need to use an obsolete version of the Internet Key Exchange in order to achieve that security goal.

The general idea is that we add an additional secret that is shared between the initiator and the responder; this secret is in addition to the authentication method that is already provided within IKEv2. We stir this secret into the SK_d value, which is used to generate the key material (KEYMAT) and the SKEYSEED for the child SAs; this secret provides quantum resistance to the IPsec SAs (and any child IKE SAs). We also stir the secret into the SK_pi, SK_pr values; this allows both sides to detect a secret mismatch cleanly.

It was considered important to minimize the changes to IKEv2. The existing mechanisms to do authentication and key exchange remain in place (that is, we continue to do (EC)DH, and potentially PKI authentication if configured). This document does not replace the authentication checks that the protocol does; instead, it is done as a parallel check.

1.1. Changes

RFC EDITOR PLEASE DELETE THIS SECTION.

Changes in this draft in each version iterations.

draft-ietf-ipsecme-qr-ikev2-04

- Using Group PPK is clarified based on comment from Quynh Dang.

draft-ietf-ipsecme-qr-ikev2-03

- Editorial changes and minor text nit fixes.
- Integrated Tommy P. text suggestions.

draft-ietf-ipsecme-qr-ikev2-02

- Added note that the PPK is stirred in the initial IKE SA setup only.
- Added note about the initiator ignoring any content in the PPK_IDENTITY notification from the responder.
- fixed Tero’s suggestions from 2/6/1028

- Added IANA assigned message types where necessary.
- fixed minor text nits

draft-ietf-ipsecme-qr-ikev2-01

- Nits and minor fixes.
- prf is replaced with prf+ for the SK_d and SK_pi/r calculations.
- Clarified using PPK in case of EAP authentication.
- PPK_SUPPORT notification is changed to USE_PPK to better reflect its purpose.

draft-ietf-ipsecme-qr-ikev2-00

- Migrated from draft-fluhrer-qr-ikev2-05 to draft-ietf-ipsecme-qr-ikev2-00 that is a WG item.

draft-fluhrer-qr-ikev2-05

- Nits and editorial fixes.
- Made PPK_ID format and PPK Distributions subsection of the PPK section. Also added an Operational Considerations section.
- Added comment about Child SA rekey in the Security Considerations section.
- Added NO_PPK_AUTH to solve the cases where a PPK_ID is not configured for a responder.
- Various text changes and clarifications.
- Expanded Security Considerations section to describe some security concerns and how they should be addressed.

draft-fluhrer-qr-ikev2-03

- Modified how we stir the PPK into the IKEv2 secret state.
- Modified how the use of PPKs is negotiated.

draft-fluhrer-qr-ikev2-02

- Simplified the protocol by stirring in the preshared key into the child SAs; this avoids the problem of having the responder decide

which preshared key to use (as it knows the initiator identity at that point); it does mean that someone with a Quantum Computer can recover the initial IKE negotiation.

- Removed positive endorsements of various algorithms. Retained warnings about algorithms known to be weak against a Quantum Computer.

draft-fluhrer-qr-ikev2-01

- Added explicit guidance as to what IKE and IPsec algorithms are quantum resistant.

draft-fluhrer-qr-ikev2-00

- We switched from using vendor ID's to transmit the additional data to notifications.

- We added a mandatory cookie exchange to allow the server to communicate to the client before the initial exchange.

- We added algorithm agility by having the server tell the client what algorithm to use in the cookie exchange.

- We have the server specify the PPK Indicator Input, which allows the server to make a trade-off between the efficiency for the search of the clients PPK, and the anonymity of the client.

- We now use the negotiated PRF (rather than a fixed HMAC-SHA256) to transform the nonces during the KDF.

1.2. Requirements Language

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].

2. Assumptions

We assume that each IKE peer has a list of Postquantum Preshared Keys (PPK) along with their identifiers (PPK_ID), and any potential IKE initiator has a selection of which PPK to use with any specific responder. In addition, implementations have a configurable flag that determines whether this postquantum preshared key is mandatory. This PPK is independent of the preshared key (if any) that the IKEv2 protocol uses to perform authentication. The PPK specific configuration that is assumed on each peer consists of the following tuple:

Peer, PPK, PPK_ID, mandatory_or_not

3. Exchanges

If the initiator is configured to use a postquantum preshared key with the responder (whether or not the use of the PPK is mandatory), then he will include a notification USE_PPK in the IKE_SA_INIT request message as follows:

```
Initiator                       Responder
-------------------------------------------------------------------
HDR, SAi1, KEi, Ni, N(USE_PPK)  --->
```

N(USE_PPK) is a status notification payload with the type 16435; it has a protocol ID of 0, no SPI and no notification data associated with it.

If the initiator needs to resend this initial message with a cookie (because the responder response included a COOKIE notification), then the resend would include the USE_PPK notification if the original message did.

If the responder does not support this specification or does not have any PPK configured, then she ignores the received notification and continues with the IKEv2 protocol as normal. Otherwise the responder checks if she has a PPK configured, and if she does, then the responder replies with the IKE_SA_INIT message including a USE_PPK notification in the response:

```
Initiator                       Responder
-------------------------------------------------------------------
                <--- HDR, SAr1, KEr, Nr, [CERTREQ], N(USE_PPK)
```

When the initiator receives this reply, he checks whether the responder included the USE_PPK notification. If the responder did not and the flag mandatory_or_not indicates that using PPKs is mandatory for communication with this responder, then the initiator MUST abort the exchange. This situation may happen in case of misconfiguration, when the initiator believes he has a mandatory to use PPK for the responder, while the responder either doesn't support PPKs at all or doesn't have any PPK configured for the initiator. See Section 6 for discussion of the possible impacts of this situation.

If the responder did not include the USE_PPK notification and using a PPK for this particular responder is optional, then the initiator continues with the IKEv2 protocol as normal, without using PPKs.

If the responder did include the USE_PPK notification, then the initiator selects a PPK, along with its identifier PPK_ID. Then, she computes this modification of the standard IKEv2 key derivation:

```
SKEYSEED = prf(Ni | Nr, g^ir)
{SK_d' | SK_ai | SK_ar | SK_ei | SK_er | SK_pi' | SK_pr' )
                = prf+ (SKEYSEED, Ni | Nr | SPIi | SPIr }

SK_d  = prf+ (PPK, SK_d')
SK_pi = prf+ (PPK, SK_pi')
SK_pr = prf+ (PPK, SK_pr')
```

That is, we use the standard IKEv2 key derivation process except that the three subkeys SK_d, SK_pi, SK_pr are run through the prf+ again, this time using the PPK as the key. Using prf+ construction ensures that it is always possible to get the resulting keys of the same size as the initial ones, even if the underlying PRF has output size different from its key size. Note, that at the time this document was written, all PRFs defined for use in IKEv2 [IKEV2-IANA-PRFS] had output size equal to the (preferred) key size. For such PRFs only the first iteration of prf+ is needed:

```
SK_d  = prf (PPK, SK_d'  | 0x01)
SK_pi = prf (PPK, SK_pi' | 0x01)
SK_pr = prf (PPK, SK_pr' | 0x01)
```

Note that the PPK is used in SK_d, SK_pi and SK_pr calculation only during the initial IKE SA setup. It MUST NOT be used when these subkeys are calculated as result of IKE SA rekey, resumption or other similar operation.

The initiator then sends the IKE_AUTH request message, including the PPK_ID value as follows:

```
Initiator                       Responder
-------------------------------------------------------------------
HDR, SK {IDi, [CERT,] [CERTREQ,]
    [IDr,] AUTH, SAi2,
    TSi, TSr, N(PPK_IDENTITY, PPK_ID), [N(NO_PPK_AUTH)]}  --->
```

PPK_IDENTITY is a status notification with the type 16436; it has a protocol ID of 0, no SPI and a notification data that consists of the identifier PPK_ID.

A situation may happen when the responder has some PPKs, but doesn't have a PPK with the PPK_ID received from the initiator. In this case the responder cannot continue with PPK (in particular, she cannot authenticate the initiator), but she could be able to continue with

normal IKEv2 protocol if the initiator provided its authentication data computed as in normal IKEv2, without using PPKs. For this purpose, if using PPKs for communication with this responder is optional for the initiator, then the initiator MAY include a notification NO_PPK_AUTH in the above message.

NO_PPK_AUTH is a status notification with the type 16437; it has a protocol ID of 0 and no SPI. The Notification Data field contains the initiator's authentication data computed using SK_pi', which has been computed without using PPKs. This is the same data that would normally be placed in the Authentication Data field of an AUTH payload. Since the Auth Method field is not present in the notification, the authentication method used for computing the authentication data MUST be the same as method indicated in the AUTH payload. Note that if the initiator decides to include the NO_PPK_AUTH notification, the initiator needs to perform authentication data computation twice, which may consume computation power (e.g. if digital signatures are involved).

When the responder receives this encrypted exchange, she first computes the values:

```
 SKEYSEED = prf(Ni | Nr, g^ir)
 {SK_d' | SK_ai | SK_ar | SK_ei | SK_er | SK_pi' | SK_pr' }
                  = prf+ (SKEYSEED, Ni | Nr | SPIi | SPIr )
```

She then uses the SK_ei/SK_ai values to decrypt/check the message and then scans through the payloads for the PPK_ID attached to the PPK_IDENTITY notification. If no PPK_IDENTITY notification is found and the peers successfully exchanged USE_PPK notifications in the IKE_SA_INIT exchange, then the responder MUST send back AUTHENTICATION_FAILED notification and then fail the negotiation.

If the PPK_IDENTITY notification contains PPK_ID that is not known to the responder or is not configured for use for the identity from IDi payload, then the responder checks whether using PPKs for this initiator is mandatory and whether the initiator included NO_PPK_AUTH notification in the message. If using PPKs is mandatory or no NO_PPK_AUTH notification found, then then the responder MUST send back AUTHENTICATION_FAILED notification and then fail the negotiation. Otherwise (when PPK is optional and the initiator included NO_PPK_AUTH notification) the responder MAY continue regular IKEv2 protocol, except that she uses the data from the NO_PPK_AUTH notification as the authentication data (which usually resides in the AUTH payload), for the purpose of the initiator authentication. Note, that Authentication Method is still indicated in the AUTH payload.

This table summarizes the above logic for the responder:

Received USE_PPK	Received NO_PPK_AUTH	Have PPK	PPK Mandatory	Action
No	*	No	*	Standard IKEv2 protocol
No	*	Yes	No	Standard IKEv2 protocol
No	*	Yes	Yes	Abort negotiation
Yes	No	No	*	Abort negotiation
Yes	Yes	No	Yes	Abort negotiation
Yes	Yes	No	No	Standard IKEv2 protocol
Yes	*	Yes	*	Use PPK

If PPK is in use, then the responder extracts the corresponding PPK and computes the following values:

```
SK_d  = prf+ (PPK, SK_d')
SK_pi = prf+ (PPK, SK_pi')
SK_pr = prf+ (PPK, SK_pr')
```

The responder then continues with the IKE_AUTH exchange (validating the AUTH payload that the initiator included) as usual and sends back a response, which includes the PPK_IDENTITY notification with no data to indicate that the PPK is used in the exchange:

```
Initiator                       Responder
-----------------------------------------------------------------
                          <--   HDR, SK {IDr, [CERT,]
                                AUTH, SAr2,
                                TSi, TSr, N(PPK_IDENTITY)}
```

When the initiator receives the response, then he checks for the presence of the PPK_IDENTITY notification. If he receives one, he marks the SA as using the configured PPK to generate SK_d, SK_pi, SK_pr (as shown above); the content of the received PPK_IDENTITY (if any) MUST be ignored. If the initiator does not receive the PPK_IDENTITY, he MUST either fail the IKE SA negotiation sending the AUTHENTICATION_FAILED notification in the Informational exchange (if the PPK was configured as mandatory), or continue without using the PPK (if the PPK was not configured as mandatory and the initiator included the NO_PPK_AUTH notification in the request).

If EAP is used in the IKE_AUTH exchange, then the initiator doesn't include AUTH payload in the first request message, however the responder sends back AUTH payload in the first reply. The peers then exchange AUTH payloads after EAP is successfully completed. As a result, the responder sends AUTH payload twice - in the first IKE_AUTH reply message and in the last one, while the initiator sends

AUTH payload only in the last IKE_AUTH request. See more details about EAP authentication in IKEv2 in Section 2.16 of [RFC7296].

The general rule for using PPK in the IKE_AUTH exchange, which covers EAP authentication case too, is that the initiator includes PPK_IDENTITY (and optionally NO_PPK_AUTH) notification in the request message containing AUTH payload. Therefore, in case of EAP the responder always computes the AUTH payload in the first IKE_AUTH reply message without using PPK (by means of SK_pr'), since PPK_ID is not yet known to the responder. Once the IKE_AUTH request message containing PPK_IDENTITY notification is received, the responder follows rules described above for non-EAP authentication case.

```
Initiator                           Responder
-----------------------------------------------------------------
HDR, SK {IDi, [CERTREQ,]
    [IDr,] SAi2,
    TSi, TSr}  -->
                                <--  HDR, SK {IDr, [CERT,] AUTH,
                                         EAP}
HDR, SK {EAP}  -->
                                <--  HDR, SK {EAP (success)}
HDR, SK {AUTH,
    N(PPK_IDENTITY, PPK_ID)
    [, N(NO_PPK_AUTH)]}  -->
                                <--  HDR, SK {AUTH, SAr2, TSi, TSr
                                     [, N(PPK_IDENTITY)]}
```

Note, that the IKE_SA_INIT exchange in case of PPK is as described above (including exchange of the USE_PPK notifications), regardless whether EAP is employed in the IKE_AUTH or not.

4. Upgrade procedure

This algorithm was designed so that someone can introduce PPKs into an existing IKE network without causing network disruption.

In the initial phase of the network upgrade, the network administrator would visit each IKE node, and configure:

- The set of PPKs (and corresponding PPK_IDs) that this node would need to know.
- For each peer that this node would initiate to, which PPK will be used.
- That the use of PPK is currently not mandatory.

With this configuration, the node will continue to operate with nodes that have not yet been upgraded. This is due to the USE_PPK notify and the NO_PPK_AUTH notify; if the initiator has not been upgraded, he will not send the USE_PPK notify (and so the responder will know that we will not use a PPK). If the responder has not been upgraded, she will not send the USE_PPK notify (and so the initiator will know to not use a PPK). If both peers have been upgraded, but the responder isn't yet configured with the PPK for the initiator, then the responder could do standard IKEv2 protocol if the initiator sent NO_PPK_AUTH notification. If both the responder and initiator have been upgraded and properly configured, they will both realize it, and in that case, the link will be quantum secure.

As an optional second step, after all nodes have been upgraded, then the administrator may then go back through the nodes, and mark the use of PPK as mandatory. This will not affect the strength against a passive attacker; it would mean that an attacker with a Quantum Computer (which is sufficiently fast to be able to break the (EC)DH in real time) would not be able to perform a downgrade attack.

5. PPK

5.1. PPK_ID format

This standard requires that both the initiator and the responder have a secret PPK value, with the responder selecting the PPK based on the PPK_ID that the initiator sends. In this standard, both the initiator and the responder are configured with fixed PPK and PPK_ID values, and do the look up based on PPK_ID value. It is anticipated that later standards will extend this technique to allow dynamically changing PPK values. To facilitate such an extension, we specify that the PPK_ID the initiator sends will have its first octet be the PPK_ID Type value. This document defines two values for PPK_ID Type:

- PPK_ID_OPAQUE (1) - for this type the format of the PPK_ID (and the PPK itself) is not specified by this document; it is assumed to be mutually intelligible by both by initiator and the responder. This PPK_ID type is intended for those implementations that choose not to disclose the type of PPK to active attackers.

- PPK_ID_FIXED (2) - in this case the format of the PPK_ID and the PPK are fixed octet strings; the remaining bytes of the PPK_ID are a configured value. We assume that there is a fixed mapping between PPK_ID and PPK, which is configured locally to both the initiator and the responder. The responder can use to do a look up the passed PPK_ID value to determine the corresponding PPK value. Not all implementations are able to configure arbitrary octet strings; to improve the potential interoperability, it is

recommended that, in the PPK_ID_FIXED case, both the PPK and the PPK_ID strings be limited to the base64 character set, namely the 64 characters 0-9, A-Z, a-z, + and /.

The PPK_ID type value 0 is reserved; values 3-127 are reserved for IANA; values 128-255 are for private use among mutually consenting parties.

5.2. Operational Considerations

The need to maintain several independent sets of security credentials can significantly complicate a security administrator's job, and can potentially slow down widespread adoption of this specification. It is anticipated, that administrators will try to simplify their job by decreasing the number of credentials they need to maintain. This section describes some of the considerations for PPK management.

5.2.1. PPK Distribution

PPK_IDs of the type PPK_ID_FIXED (and the corresponding PPKs) are assumed to be configured within the IKE device in an out-of-band fashion. While the method of distribution is a local matter and out of scope of this document or IKEv2, [RFC6030] describes a format for symmetric key exchange. That format could be reused with the Key Id field being the PPK_ID (without the PPK_ID Type octet for a PPK_ID_FIXED), the PPK being the secret, and algorithm ("Algorithm=urn:ietf:params:xml:ns:keyprov:pskc:pin") as the PIN.

5.2.2. Group PPK

This document doesn't explicitly require that PPK is unique for each pair of peers. If it is the case, then this solution provides full peer authentication, but it also means that each host must have as many independent PPKs as the peers it is going to communicate with. As the number of peers grows the PPKs will not scale.

It is possible to use a single PPK for a group of users. Since each peer uses classical public key cryptography in addition to PPK for key exchange and authentication, members of the group can neither impersonate each other nor read other's traffic, unless they use Quantum Computers to break public key operations. However group members can record other members' traffic and decrypt it later, when they get access to a Quantum Computer.

In addition, the fact that the PPK is known to a (potentially large) group of users makes it more susceptible to theft. When an attacker equipped with a Quantum Computer got access to a group PPK, all communications inside the group are revealed.

For these reasons using group PPK is NOT RECOMMENDED.

5.2.3. PPK-only Authentication

If Quantum Computers become a reality, classical public key cryptography will provide little security, so administrators may find it attractive not to use it at all for authentication. This will reduce the number of credentials they need to maintain to PPKs only. Combining group PPK and PPK-only authentication is NOT RECOMMENDED, since in this case any member of the group can impersonate any other member even without help of Quantum Computers.

PPK-only authentication can be achieved in IKEv2 if NULL Authentication method [RFC7619] is employed. Without PPK the NULL Authentication method provides no authentication of the peers, however since a PPK is stirred into the SK_pi and the SK_pr, the peers become authenticated if a PPK is in use. Using PPKs MUST be mandatory for the peers if they advertise support for PPK in IKE_SA_INIT and use NULL Authentication. Addtionally, since the peers are authenticated via PPK, the ID Type in the IDi/IDr payloads SHOULD NOT be ID_NULL, despite using the NULL Authentication method.

6. Security Considerations

Quantum computers are able to perform Grover's algorithm; that effectively halves the size of a symmetric key. Because of this, the user SHOULD ensure that the postquantum preshared key used has at least 256 bits of entropy, in order to provide 128-bit security level.

With this protocol, the computed SK_d is a function of the PPK. Assuming that the PPK has sufficient entropy (for example, at least 2^256 possible values), then even if an attacker was able to recover the rest of the inputs to the PRF function, it would be infeasible to use Grover's algorithm with a Quantum Computer to recover the SK_d value. Similarly, every child SA key is a function of SK_d, hence all the keys for all the child SAs are also quantum resistant (assuming that the PPK was of high enough entropy, and that all the subkeys are sufficiently long).

Although this protocol preserves all the security properties of IKEv2 against adversaries with conventional computers, it allows an adversary with a Quantum Computer to decrypt all traffic encrypted with the initial IKE SA. In particular, it allows the adversary to recover the identities of both sides. If there is IKE traffic other than the identities that need to be protected against such an adversary, implementations MAY rekey the initial IKE SA immediately after negotiating it to generate a new SKEYSEED from the postquantum

SK_d. This would reduce the amount of data available to an attacker with a Quantum Computer.

If sensitive information (like keys) is to be transferred over IKE SA, then implementations MUST rekey the initial IKE SA before sending this information to get protection against Quantum Computers.

Alternatively, an initial IKE SA (which is used to exchange identities) can take place, perhaps by using the protocol documented in [RFC6023]. After the childless IKE SA is created, implementations would immediately create a new IKE SA (which is used to exchange everything else) by using a rekey mechanism for IKE SAs. Because the rekeyed IKE SA keys are a function of SK_d, which is a function of the PPK (among other things), traffic protected by that IKE SA is secure against Quantum capable adversaries.

In addition, the policy SHOULD be set to negotiate only quantum-resistant symmetric algorithms; while this RFC doesn't claim to give advice as to what algorithms are secure (as that may change based on future cryptographical results), below is a list of defined IKEv2 and IPsec algorithms that should NOT be used, as they are known not to be quantum resistant

- Any IKEv2 Encryption algorithm, PRF or Integrity algorithm with key size less than 256 bits.

- Any ESP Transform with key size less than 256 bits.

- PRF_AES128_XCBC and PRF_AES128_CBC; even though they are defined to be able to use an arbitrary key size, they convert it into a 128-bit key internally.

Section 3 requires the initiator to abort the initial exchange if using PPKs is mandatory for it, but the responder might not include the USE_PPK notification in the response. In this situation when the initiator aborts negotiation he leaves half-open IKE SA on the responder (because IKE_SA_INIT completes successfully from the responder's point of view). This half-open SA will eventually expire and be deleted, but if the initiator continues its attempts to create IKE SA with a high enough rate, then the responder may consider it as a Denial-of-Service attack and take protection measures (see [RFC8019] for more detail). It is RECOMMENDED that implementations in this situation cache the negative result of negotiation for some time and don't make attempts to create it again for some time, because this is a result of misconfiguration and probably some re-configuration of the peers is needed.

If using PPKs is optional for both peers and they authenticate themselves using digital signatures, then an attacker in between, equipped with a Quantum Computer capable of breaking public key operations in real time, is able to mount downgrade attack by removing USE_PPK notification from the IKE_SA_INIT and forging digital signatures in the subsequent exchange. If using PPKs is mandatory for at least one of the peers or PSK is used for authentication, then the attack will be detected and the SA won't be created.

If using PPKs is mandatory for the initiator, then an attacker capable to eavesdrop and to inject packets into the network can prevent creating IKE SA by mounting the following attack. The attacker intercepts the initial request containing the USE_PPK notification and injects the forget response containing no USE_PPK. If the attacker manages to inject this packet before the responder sends a genuine response, then the initiator would abort the exchange. To thwart this kind of attack it is RECOMMENDED, that if using PPKs is mandatory for the initiator and the received response doesn't contain the USE_PPK notification, then the initiator doesn't abort the exchange immediately, but instead waits some time for more responses (possibly retransmitting the request). If all the received responses contain no USE_PPK, then the exchange is aborted.

7. IANA Considerations

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

16435	USE_PPK
16436	PPK_IDENTITY
16437	NO_PPK_AUTH

This document also creates a new IANA registry for the PPK_ID types. The initial values of this registry are:

PPK_ID Type	Value
Reserved	0
PPK_ID_OPAQUE	1
PPK_ID_FIXED	2
Unassigned	3-127
Reserved for private use	128-255

Changes and additions to this registry are by Expert Review [RFC5226].

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>.

8.2. Informational References

[I-D.hoffman-c2pq] Hoffman, P., "The Transition from Classical to Post-Quantum Cryptography", draft-hoffman-c2pq-03 (work in progress), February 2018.

[IKEV2-IANA-PRFS] "Internet Key Exchange Version 2 (IKEv2) Parameters, Transform Type 2 - Pseudorandom Function Transform IDs", <https://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml#ikev2-parameters-6>.

[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>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <https://www.rfc-editor.org/info/rfc5226>.

[RFC6023] Nir, Y., Tschofenig, H., Deng, H., and R. Singh, "A Childless Initiation of the Internet Key Exchange Version 2 (IKEv2) Security Association (SA)", RFC 6023, DOI 10.17487/RFC6023, October 2010, <https://www.rfc-editor.org/info/rfc6023>.

[RFC6030] Hoyer, P., Pei, M., and S. Machani, "Portable Symmetric Key Container (PSKC)", RFC 6030, DOI 10.17487/RFC6030, October 2010, <https://www.rfc-editor.org/info/rfc6030>.

[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>.

[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>.

Appendix A. Discussion and Rationale

The idea behind this document is that while a Quantum Computer can easily reconstruct the shared secret of an (EC)DH exchange, they cannot as easily recover a secret from a symmetric exchange. This makes the SK_d, and hence the IPsec KEYMAT and any child SA's SKEYSEED, depend on both the symmetric PPK, and also the Diffie-Hellman exchange. If we assume that the attacker knows everything except the PPK during the key exchange, and there are 2^n plausible PPKs, then a Quantum Computer (using Grover's algorithm) would take O(2^(n/2)) time to recover the PPK. So, even if the (EC)DH can be trivially solved, the attacker still can't recover any key material (except for the SK_ei, SK_er, SK_ai, SK_ar values for the initial IKE exchange) unless they can find the PPK, which is too difficult if the PPK has enough entropy (for example, 256 bits). Note that we do allow an attacker with a Quantum Computer to rederive the keying material for the initial IKE SA; this was a compromise to allow the responder to select the correct PPK quickly.

Another goal of this protocol is to minimize the number of changes within the IKEv2 protocol, and in particular, within the cryptography of IKEv2. By limiting our changes to notifications, and adjusting the SK_d, SK_pi, SK_pr, it is hoped that this would be implementable, even on systems that perform most of the IKEv2 processing in hardware.

A third goal was to be friendly to incremental deployment in operational networks, for which we might not want to have a global shared key, or quantum resistant IKEv2 is rolled out incrementally. This is why we specifically try to allow the PPK to be dependent on the peer, and why we allow the PPK to be configured as optional.

A fourth goal was to avoid violating any of the security goals of IKEv2.

Appendix B. Acknowledgements

We would like to thank Tero Kivinen, Paul Wouters, Graham Bartlett, Tommy Pauly, Quynh Dang and the rest of the IPSecME Working Group for their feedback and suggestions for the scheme.

Authors' Addresses

Scott Fluhrer
Cisco Systems

Email: sfluhrer@cisco.com

David McGrew
Cisco Systems

Email: mcgrew@cisco.com

Panos Kampanakis
Cisco Systems

Email: pkampana@cisco.com

Valery Smyslov
ELVIS-PLUS

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

Network
Internet-Draft
Intended status: Standards Track
Expires: May 7, 2019

T. Pauly
Apple Inc.
P. Wouters
Red Hat

November 3, 2018

Split DNS Configuration for IKEv2
draft-ietf-ipsecme-split-dns-14

Abstract

This document defines two Configuration Payload Attribute Types for the IKEv2 protocol that add support for private DNS domains. These domains are intended to be resolved using DNS servers reachable through an IPsec connection, while leaving all other DNS resolution unchanged. This approach of resolving a subset of domains using non-public DNS servers is referred to as "Split DNS".

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 7, 2019.

Copyright Notice

Copyright (c) 2018 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

Split DNS is a common configuration for secure tunnels, such as Virtual Private Networks in which host machines private to an organization can only be resolved using internal DNS resolvers [RFC2775]. In such configurations, it is often desirable to only resolve hosts within a set of private domains using the tunnel, while letting resolutions for public hosts be handled by a device's default DNS configuration.

The Internet Key Exchange protocol version 2 [RFC7296] negotiates configuration parameters using Configuration Payload Attribute Types. This document defines two Configuration Payload Attribute Types that add support for trusted Split DNS domains.

The INTERNAL_DNS_DOMAIN attribute type is used to convey one or more DNS domains that MUST be resolved only using the provided DNS nameserver IP addresses, causing these requests to use the IPsec connection.

The INTERNAL_DNSSEC_TA attribute type is used to convey DNSSEC trust anchors for those domains.

When only a subset of traffic is routed into a private network using an IPsec SA, these Configuration Payload options can be used to define which private domains are intended to be resolved through the IPsec connection without affecting the client's global DNS resolution.

For the purposes of this document, DNS resolution servers accessible through an IPsec connection will be referred to as "internal DNS servers", and other DNS servers will be referred to as "external DNS servers".

A client using these configuration payloads will be able to request and receive Split DNS configurations using the INTERNAL_DNS_DOMAIN and INTERNAL_DNSSEC_TA configuration attributes. The client device can use the internal DNS server(s) for any DNS queries within the assigned domains. DNS queries for other domains SHOULD be sent to the regular external DNS server.

Other tunnel-establishment protocols already support the assignment of Split DNS domains. For example, there are proprietary extensions to IKEv1 that allow a server to assign Split DNS domains to a client. However, the IKEv2 standard does not include a method to configure this option. This document defines a standard way to negotiate this option for IKEv2.

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 captials, as shown here.

2. Protocol Exchange

In order to negotiate which domains are considered internal to an IKEv2 tunnel, initiators indicate support for Split DNS in their CFG_REQUEST payloads, and responders assign internal domains (and DNSSEC trust anchors) in their CFG_REPLY payloads. When Split DNS has been negotiated, the existing DNS server configuration attributes will be interpreted as internal DNS servers that can resolve hostnames within the internal domains.

2.1. Configuration Request

To indicate support for Split DNS, an initiator includes one more INTERNAL_DNS_DOMAIN attributes as defined in Section 3 as part of the CFG_REQUEST payload. If an INTERNAL_DNS_DOMAIN attribute is included in the CFG_REQUEST, the initiator MUST also include one or more INTERNAL_IP4_DNS and INTERNAL_IP6_DNS attributes in the CFG_REQUEST.

The INTERNAL_DNS_DOMAIN attribute sent by the initiator is usually empty but MAY contain a suggested domain name.

The absence of INTERNAL_DNS_DOMAIN attributes in the CFG_REQUEST payload indicates that the initiator does not support or is unwilling to accept Split DNS configuration.

To indicate support for DNSSEC, an initiator includes one or more INTERNAL_DNSSEC_TA attributes as defined in Section 3 as part of the CFG_REQUEST payload. If an INTERNAL_DNSSEC_TA attribute is included in the CFG_REQUEST, the initiator MUST also include one or more INTERNAL_DNS_DOMAIN attributes in the CFG_REQUEST. If the initiator includes an INTERNAL_DNSSEC_TA attribute, but does not inclue an INTERNAL_DNS_DOMAIN attribute, the responder MAY still respond with both INTERNAL_DNSSEC_TA and INTERNAL_DNS_DOMAIN attributes.

An initiator MAY convey its current DNSSEC trust anchors for the domain specified in the INTERNAL_DNS_DOMAIN attribute. If it does not wish to convey this information, it MUST use a length of 0.

The absence of INTERNAL_DNSSEC_TA attributes in the CFG_REQUEST payload indicates that the initiator does not support or is unwilling to accept DNSSEC trust anchor configuration.

2.2. Configuration Reply

Responders MAY send one or more INTERNAL_DNS_DOMAIN attributes in their CFG_REPLY payload. If an INTERNAL_DNS_DOMAIN attribute is included in the CFG_REPLY, the responder MUST also include one or both of the INTERNAL_IP4_DNS and INTERNAL_IP6_DNS attributes in the CFG_REPLY. These DNS server configurations are necessary to define which servers can receive queries for hostnames in internal domains. If the CFG_REQUEST included an INTERNAL_DNS_DOMAIN attribute, but the CFG_REPLY does not include an INTERNAL_DNS_DOMAIN attribute, the initiator MUST behave as if Split DNS configurations are not supported by the server, unless the initiator has been configured with local polict to define a set of Split DNS domains to use by default.

Each INTERNAL_DNS_DOMAIN represents a domain that the DNS servers address listed in INTERNAL_IP4_DNS and INTERNAL_IP6_DNS can resolve.

If the CFG_REQUEST included INTERNAL_DNS_DOMAIN attributes with non-zero lengths, the content MAY be ignored or be interpreted as a suggestion by the responder.

For each DNS domain specified in an INTERNAL_DNS_DOMAIN attribute, one or more INTERNAL_DNSSEC_TA attributes MAY be included by the responder. This attribute lists the corresponding internal DNSSEC trust anchor in the DNS presentation format of a DS record as specified in [RFC4034]. The INTERNAL_DNSSEC_TA attribute MUST immediately follow the INTERNAL_DNS_DOMAIN attribute that it applies to.

2.3. Mapping DNS Servers to Domains

All DNS servers provided in the CFG_REPLY MUST support resolving hostnames within all INTERNAL_DNS_DOMAIN domains. In other words, the INTERNAL_DNS_DOMAIN attributes in a CFG_REPLY payload form a single list of Split DNS domains that applies to the entire list of INTERNAL_IP4_DNS and INTERNAL_IP6_DNS attributes.

2.4. Example Exchanges

2.4.1. Simple Case

In this example exchange, the initiator requests INTERNAL_IP4_DNS and INTERNAL_DNS_DOMAIN attributes in the CFG_REQUEST, but does not specify any value for either. This indicates that it supports Split DNS, but has no preference for which DNS requests will be routed through the tunnel.

The responder replies with two DNS server addresses, and two internal domains, "example.com" and "city.other.com".

Any subsequent DNS queries from the initiator for domains such as "www.example.com" SHOULD use 198.51.100.2 or 198.51.100.4 to resolve.

```
CP(CFG_REQUEST) =
  INTERNAL_IP4_ADDRESS()
  INTERNAL_IP4_DNS()
  INTERNAL_DNS_DOMAIN()

CP(CFG_REPLY) =
  INTERNAL_IP4_ADDRESS(198.51.100.234)
  INTERNAL_IP4_DNS(198.51.100.2)
  INTERNAL_IP4_DNS(198.51.100.4)
  INTERNAL_DNS_DOMAIN(example.com)
  INTERNAL_DNS_DOMAIN(city.other.com)
```

2.4.2. Requesting Domains and DNSSEC trust anchors

In this example exchange, the initiator requests INTERNAL_IP4_DNS, INTERNAL_DNS_DOMAIN and INTERNAL_DNSSEC_TA attributes in the CFG_REQUEST.

Any subsequent DNS queries from the initiator for domains such as "www.example.com" or "city.other.com" would be DNSSEC validated using the DNSSEC trust anchor received in the CFG_REPLY.

In this example, the initiator has no existing DNSSEC trust anchors would the requested domain. the "example.com" dommain has DNSSEC trust anchors that are returned, while the "other.com" domain has no DNSSEC trust anchors.

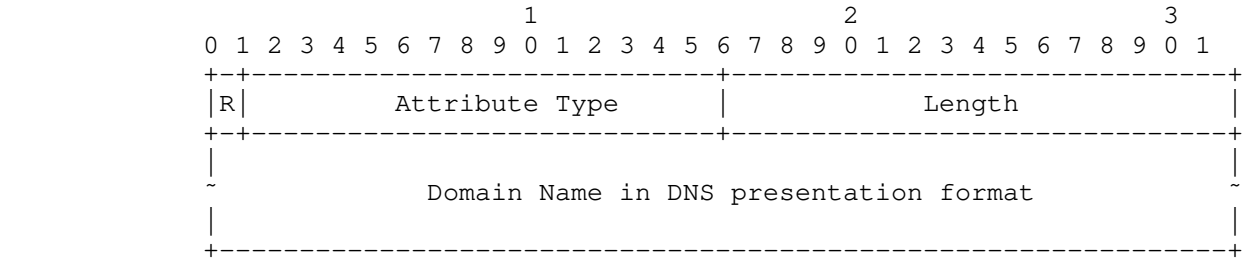
```
CP(CFG_REQUEST) =
  INTERNAL_IP4_ADDRESS()
  INTERNAL_IP4_DNS()
  INTERNAL_DNS_DOMAIN()
  INTERNAL_DNSSEC_TA()

CP(CFG_REPLY) =
  INTERNAL_IP4_ADDRESS(198.51.100.234)
  INTERNAL_IP4_DNS(198.51.100.2)
  INTERNAL_IP4_DNS(198.51.100.4)
  INTERNAL_DNS_DOMAIN(example.com)
  INTERNAL_DNSSEC_TA(43547,8,1,B6225AB2CC613E0DCA7962BDC2342EA4...)
  INTERNAL_DNSSEC_TA(31406,8,2,F78CF3344F72137235098ECBBD08947C...)
  INTERNAL_DNS_DOMAIN(city.other.com)
```

3. Payload Formats

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

3.1. INTERNAL_DNS_DOMAIN Configuration Attribute Type Request and Reply

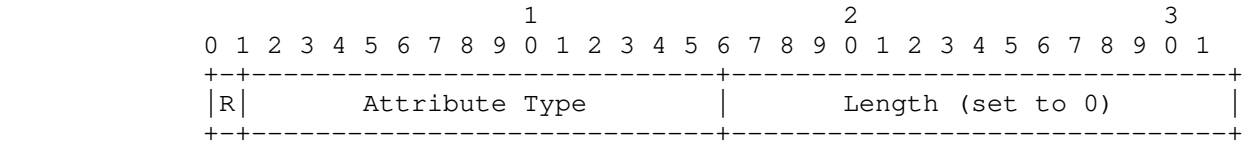


- Reserved (1 bit) - Defined in IKEv2 RFC [RFC7296].
- Attribute Type (15 bits) set to value 25 for INTERNAL_DNS_DOMAIN.
- Length (2 octets) - Length of domain name.
- Domain Name (0 or more octets) - A Fully Qualified Domain Name used for Split DNS rules, such as "example.com", in DNS presentation format and optionally using IDNA [RFC5890] for Internationalized Domain Names. Implementors need to be careful that this value is not null-terminated.

3.2. INTERNAL_DNSSEC_TA Configuration Attribute

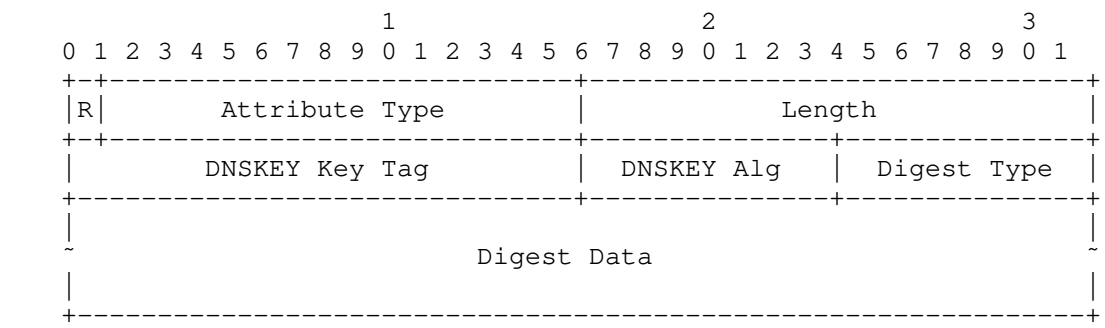
An INTERNAL_DNSSEC_TA Configuration Attribute can either be empty, or it can contain one Trust Anchor by containing a non-zero Length with a DNSKEY Key Tag, DNSKEY Algorithm, Digest Type and Digest Data fields.

An empty INTERNAL_DNSSEC_TA CFG attribute:



- Reserved (1 bit) - Defined in IKEv2 RFC [RFC7296].
- Attribute Type (15 bits) set to value 26 for INTERNAL_DNSSEC_TA.
- Length (2 octets) - Set to 0 for an empty attribute.

A non-empty INTERNAL_DNSSEC_TA CFG attribute:



- Reserved (1 bit) - Defined in IKEv2 RFC [RFC7296].

- Attribute Type (15 bits) set to value 26 for INTERNAL_DNSSEC_TA.

- Length (2 octets) - Length of DNSSEC Trust Anchor data (4 octets plus the length of the Digest Data).

- DNSKEY Key Tag value (2 octets) - Delegation Signer (DS) Key Tag as specified in [RFC4034] Section 5.1.

- DNSKEY Algorithm (1 octet) - DNSKEY algorithm value from the IANA DNS Security Algorithm Numbers Registry.

- Digest Type (1 octet) - DS algorithm value from the IANA Delegation Signer (DS) Resource Record (RR) Type Digest Algorithms Registry.

- Digest Data (1 or more octets) - The DNSKEY digest as specified in [RFC4034] Section 5.1 in presentation format.

Each INTERNAL_DNSSEC_TA attribute in the CFG_REPLY payload MUST immediately follow a corresponding INTERNAL_DNS_DOMAIN attribute. As the INTERNAL_DNSSEC_TA format itself does not contain the domain name, it relies on the preceding INTERNAL_DNS_DOMAIN to provide the domain for which it specifies the trust anchor. Any INTERNAL_DNSSEC_TA attribute that is not immediately preceded by an INTERNAL_DNS_DOMAIN or another INTERNAL_DNSSEC_TA attribute applying to the same domain name MUST be ignored and treated as a protocol error.

4. INTERNAL_DNS_DOMAIN Usage Guidelines

If a CFG_REPLY payload contains no INTERNAL_DNS_DOMAIN attributes, the client MAY use the provided INTERNAL_IP4_DNS or INTERNAL_IP6_DNS servers as the default DNS server(s) for all queries.

If a client is configured by local policy to only accept a limited number of INTERNAL_DNS_DOMAIN values, the client MUST ignore any other INTERNAL_DNS_DOMAIN values.

For each INTERNAL_DNS_DOMAIN entry in a CFG_REPLY payload that is not prohibited by local policy, the client MUST use the provided INTERNAL_IP4_DNS or INTERNAL_IP6_DNS DNS servers as the only resolvers for the listed domains and its sub-domains and it MUST NOT attempt to resolve the provided DNS domains using its external DNS servers. Other domain names SHOULD be resolved using some other external DNS resolver(s), configured independently from IKE. Queries for these other domains MAY be sent to the internal DNS resolver(s) listed in that CFG_REPLY message, but have no guarantee of being answered. For example, if the INTERNAL_DNS_DOMAIN attribute specifies "example.com", then "example.com", "www.example.com" and "mail.eng.example.com" MUST be resolved using the internal DNS resolver(s), but "anotherexample.com" and "ample.com" SHOULD NOT be resolved using the internal resolver and SHOULD use the system's external DNS resolver(s).

The initiator SHOULD allow the DNS domains listed in the INTERNAL_DNS_DOMAIN attributes to resolve to special IP address ranges, such as those of [RFC1918], even if the initiator host is otherwise configured to block DNS answer containing these special IP addresses.

When an IKE SA is terminated, the DNS forwarding MUST be unconfigured. This includes deleting the DNS forwarding rules; flushing all cached data for DNS domains provided by the INTERNAL_DNS_DOMAIN attribute, including negative cache entries; removing any obtained DNSSEC trust anchors from the list of trust anchors; and clearing the outstanding DNS request queue.

INTERNAL_DNS_DOMAIN attributes SHOULD only be used on split tunnel configurations where only a subset of traffic is routed into a private remote network using the IPsec connection. If all traffic is routed over the IPsec connection, the existing global INTERNAL_IP4_DNS and INTERNAL_IP6_DNS can be used without creating specific DNS exemptions.

5. INTERNAL_DNSSEC_TA Usage Guidelines

DNS records can be used to publish specific records containing trust anchors for applications. The most common record type is the TLSA record specified in [RFC6698]. This DNS record type publishes which CA certificate or EE certificate to expect for a certain host name. These records are protected by DNSSEC and thus can be trusted by the application. Whether to trust TLSA records instead of the traditional WebPKI depends on the local policy of the client. By accepting an INTERNAL_DNSSEC_TA trust anchor via IKE from the remote IKE server, the IPsec client might be allowing the remote IKE server to override the trusted certificates for TLS. Similar override concerns apply to other public key or fingerprint based DNS records, such as OPENPGPKEY, SMIMEA or IPSECKEY records.

Thus, installing an INTERNAL_DNSSEC_TA trust anchor can be seen as the equivalent of installing an Enterprise Certificate Agency (CA) certificate. It allows the remote IKE/IPsec server to modify DNS answers including its DNSSEC cryptographic signatures by overriding existing DNS information with trust anchor conveyed via IKE and (temporarilly) installed on the IKE client. Of specific concern is the overriding of [RFC6698] based TLSA records, which represent a confirmation or override of an existing WebPKI TLS certificate. Other DNS record types that convey cryptographic materials (public keys or fingerprints) are OPENPGPKEY, SMIMEA, SSHP and IPSECKEY records.

IKE clients willing to accept INTERNAL_DNSSEC_TA attributes MUST use a whitelist of one or more domains that can be updated out of band. IKE clients with an empty whitelist MUST NOT use any INTERNAL_DNSSEC_TA attributes received over IKE. Such clients MAY interpret receiving an INTERNAL_DNSSEC_TA attribute for a non-whitelisted domain as an indication that their local configuration may need to be updated out of band.

IKE clients should take care to only whitelist domains that apply to internal or managed domains, rather than to generic Internet traffic. The DNS root zone (".") MUST NOT be whitelisted. Other generic or public domains, such as top-level domains, similarly SHOULD NOT be whitelisted.

Any updates to this whitelist of domain names MUST happen via explicit human interaction to prevent invisible installation of trust anchors.

IKE clients SHOULD accept any INTERNAL_DNSSEC_TA updates for subdomain names of the whitelisted domain names. For example, if

"example.net" is whitelisted, then INTERNAL_DNSSEC_TA received for "antartica.example.net" SHOULD be accepted.

IKE clients MAY interpret an INTERNAL_DNSSEC_TA for domain that was not preconfigured as an indication that it needs to update its IKE configuration (out of band). The client MUST NOT use such a INTERNAL_DNSSEC_TA to reconfigure its local DNS settings.

IKE clients MUST ignore any received INTERNAL_DNSSEC_TA requests for a FDQN for which it did not receive and accept an INTERNAL_DNS_DOMAIN Configuration Payload.

In most deployment scenario's, the IKE client has an expectation that it is connecting, using a split-network setup, to a specific organisation or enterprise. A recommended policy would be to only accept INTERNAL_DNSSEC_TA directives from that organization's DNS names. However, this might not be possible in all deployment scenarios, such as one where the IKE server is handing out a number of domains that are not within one parent domain.

6. Security Considerations

The use of Split DNS configurations assigned by an IKEv2 responder is predicated on the trust established during IKE SA authentication. However, if IKEv2 is being negotiated with an anonymous or unknown endpoint (such as for Opportunistic Security [RFC7435]), the initiator MUST ignore Split DNS configurations assigned by the responder.

If a host connected to an authenticated IKE peer is connecting to another IKE peer that attempts to claim the same domain via the INTERNAL_DNS_DOMAIN attribute, the IKE connection SHOULD only process the DNS information if the two connections are part of the same logical entity. Otherwise, the client SHOULD refuse the DNS information and potentially warn the end-user.

If the initiator is using DNSSEC validation for a domain in its public DNS view, and it requests and receives an INTERNAL_DNS_DOMAIN attribute without an INTERNAL_DNSSEC_TA, it will need to reconfigure its DNS resolver to allow for an insecure delegation. It SHOULD NOT accept insecure delegations for domains that are DNSSEC signed in the public DNS view, for which it has not explicitely requested such deletation by specifying the domain specifically using a INTERNAL_DNS_DOMAIN(domain) request.

Deployments that configure INTERNAL_DNS_DOMAIN domains should pay close attention to their use of indirect reference RRtypes such as

CNAME, DNAME, MX or SRV records so that resolving works as intended when all, some, or none of the IPsec connections are established.

The content of INTERNAL_DNS_DOMAIN and INTERNAL_DNSSEC_TA may be passed to another (DNS) program for processing. As with any network input, the content SHOULD be considered untrusted and handled accordingly.

7. IANA Considerations

This document defines two new IKEv2 Configuration Payload Attribute Types, which are allocated from the "IKEv2 Configuration Payload Attribute Types" namespace.

Value	Attribute Type	Multi-Valued	Length	Reference
25	INTERNAL_DNS_DOMAIN	YES	0 or more	[this document]
26	INTERNAL_DNSSEC_TA	YES	0 or more	[this document]

Figure 1

8. References

8.1. Normative References

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <https://www.rfc-editor.org/info/rfc1918>.

[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>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <https://www.rfc-editor.org/info/rfc4034>.

[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>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <https://www.rfc-editor.org/info/rfc6698>.

[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

[RFC2775] Carpenter, B., "Internet Transparency", RFC 2775, DOI 10.17487/RFC2775, February 2000, <https://www.rfc-editor.org/info/rfc2775>.

[RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <https://www.rfc-editor.org/info/rfc7435>.

Authors' Addresses

Tommy Pauly
Apple Inc.
One Apple Park Way
Cupertino, California 95014
US

Email: tpauly@apple.com

Paul Wouters
Red Hat

Email: pwouters@redhat.com

IPSECME
Internet-Draft
Intended status: Standards Track
Expires: December 22, 2017

D. Migault, Ed.
Ericsson
T. Guggemos, Ed.
LMU Munich
Y. Nir
Dell EMC

June 20, 2017

Implicit IV for Counter-based Ciphers in IPsec
draft-mglt-ipsecme-implicit-iv-04

Abstract

IPsec ESP sends an initialization vector (IV) or nonce in each packet, adding 8 or 16 octets. Some algorithms such as AES-GCM, AES-CCM, AES-CTR and ChaCha20-Poly1305 require a unique nonce but do not require an unpredictable nonce. When using such algorithms the packet counter value can be used to generate a nonce, saving 8 octets per packet. This document describes how to do this.

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 http://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 December 22, 2017.

Copyright Notice

Copyright (c) 2017 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 (http://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. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Introduction

Counter-based AES modes of operation such as AES-CTR ([RFC3686]), AES-CCM ([RFC4309]), and AES-GCM ([RFC4106]) require the specification of an nonce for each ESP packet. The same applies for ChaCha20-Poly1305 ([RFC7634]. Currently this nonce is sent in each ESP packet ([RFC4303]). This practice is designated in this document as "explicit nonce".

In some context, such as IoT, it may be preferable to avoid carrying the extra bytes associated to the IV and instead generate it locally on each peer. The local generation of the nonce is designated in this document as "implicit IV".

The size of this nonce depends on the specific algorithm, but all of the algorithms mentioned above take an 8-octet nonce.

This document defines how to compute the nonce locally when it is implicit. It also specifies how peers agree with the Internet Key Exchange version 2 (IKEv2 - [RFC7296]) on using an implicit IV versus an explicit IV.

This document limits its scope to the algorithms mentioned above. Other algorithms with similar properties may later be defined to use this extension.

This document does not consider AES-CBC ([RFC3602]) as AES-CBC requires the IV to be unpredictable. Deriving it directly from the packet counter as described below is insecure as mentioned in Security Consideration of [RFC3602] and has led to real world chosen plain-text attack such as BEAST [BEAST].

3. Terminology

- IoT: Internet of Things.
- IV: Initialization Vector.
- Nonce: a fixed-size octet string used only once. This is similar to IV, except that in common usage there is no implication of non-predictability.

4. Implicit IV

With the algorithms listed in Section 2, the 8 byte nonce MUST NOT repeat. The binding between a ESP packet and its nonce is provided using the Sequence Number or the Extended Sequence Number. Figure 1 and Figure 2 represent the IV with a regular 4-byte Sequence Number and with an 8-byte Extended Sequence Number respectively.

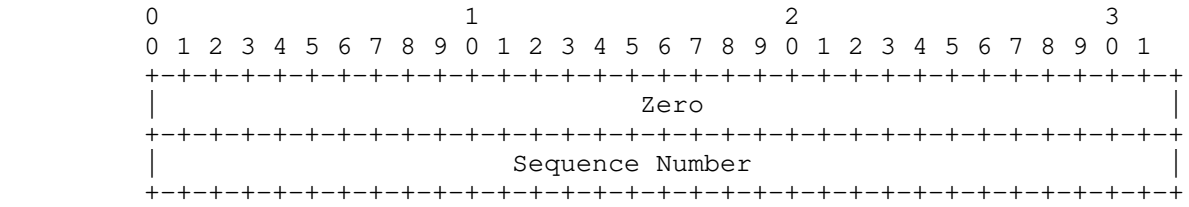


Figure 1: Implicit IV with a 4 byte Sequence Number

- Sequence Number: the 4 byte Sequence Number carried in the ESP packet.
- Zero: a 4 byte array with all bits set to zero.

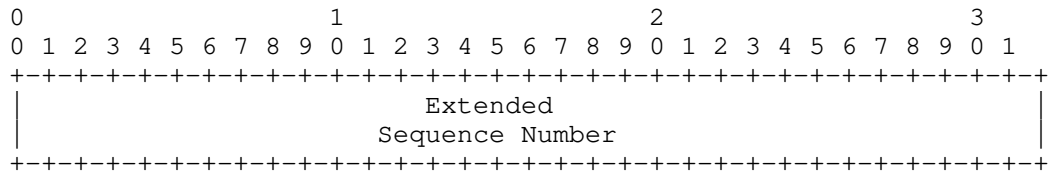


Figure 2: Implicit IV with an 8 byte Extended Sequence Number

- Extended Sequence Number: the 8 byte Extended Sequence Number of the Security Association. The 4 byte low order bytes are carried in the ESP packet.

5. Initiator Behavior

An initiator supporting this feature SHOULD propose implicit IV for all relevant algorithms. To facilitate backward compatibility with non-supporting peers the initiator SHOULD also include those same algorithms without IIV. This may require extra transforms.

6. Responder Behavior

The rules of SA payload processing ensure that the responder will never send an SA payload containing the IIV indicator to an initiator that does not support IIV.

7. Security Consideration

Nonce generation for these algorithms has not been explicitly defined. It has been left to the implementation as long as certain security requirements are met. This document provides an explicit and normative way to generate IVs. The mechanism described in this document meets the IV security requirements of all relevant algorithms.

As the IV MUST NOT repeat for one SPI when Counter-Mode ciphers are used, Implicit IV as described in this document MUST NOT be used in setups with the chance that the Sequence Number overlaps for one SPI. Multicast as described in [RFC5374], [RFC6407] and [I-D.yeung-g-ikev2] is a prominent example, where many senders share one secret and thus one SPI. Section 3.5 of [RFC6407] explains how repetition MAY BE prevented by using a prefix for each group member, which could be prefixed to the Sequence Number. Otherwise, Implicit IV MUST NOT be used in multicast scenarios.

8. IANA Considerations

AES-CTR, AES-CCM, AES-GCM and ChaCha20-Poly1305 are likely to implement the implicit IV described in this document. This section limits assignment of new code points to the recommended suites provided in [I-D.ietf-ipsecme-rfc4307bis] and [I-D.ietf-ipsecme-rfc7321bis], thus the new Transform Type 1 - Encryption Algorithm Transform IDs are as defined below:

- ENCR_AES-CCM_8_IIV
- ENCR_AES-GCM_16_IIV
- ENCR_CHACHA20-POLY1305_IIV

9. References

9.1. Normative References

[I-D.ietf-ipsecme-rfc4307bis]
Nir, Y., Kivinen, T., Wouters, P., and D. Migault, "Algorithm Implementation Requirements and Usage Guidance for IKEv2", draft-ietf-ipsecme-rfc4307bis-18 (work in progress), March 2017.

[I-D.ietf-ipsecme-rfc7321bis]
Wouters, P., Migault, D., Mattsson, J., Nir, Y., and T. Kivinen, "Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)", draft-ietf-ipsecme-rfc7321bis-06 (work in progress), June 2017.

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

[RFC3602] Frankel, S., Glenn, R., and S. Kelly, "The AES-CBC Cipher Algorithm and Its Use with IPsec", RFC 3602, DOI 10.17487/RFC3602, September 2003, <http://www.rfc-editor.org/info/rfc3602>.

[RFC3686] Housley, R., "Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)", RFC 3686, DOI 10.17487/RFC3686, January 2004, <http://www.rfc-editor.org/info/rfc3686>.

[RFC4106] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC 4106, DOI 10.17487/RFC4106, June 2005, <http://www.rfc-editor.org/info/rfc4106>.

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

[RFC4309] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", RFC 4309, DOI 10.17487/RFC4309, December 2005, <http://www.rfc-editor.org/info/rfc4309>.

[RFC5374] Weis, B., Gross, G., and D. Ignjatic, "Multicast Extensions to the Security Architecture for the Internet Protocol", RFC 5374, DOI 10.17487/RFC5374, November 2008, <http://www.rfc-editor.org/info/rfc5374>.

[RFC6407] Weis, B., Rowles, S., and T. Hardjono, "The Group Domain of Interpretation", RFC 6407, DOI 10.17487/RFC6407, October 2011, <http://www.rfc-editor.org/info/rfc6407>.

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

[RFC7634] Nir, Y., "ChaCha20, Poly1305, and Their Use in the Internet Key Exchange Protocol (IKE) and IPsec", RFC 7634, DOI 10.17487/RFC7634, August 2015, <http://www.rfc-editor.org/info/rfc7634>.

9.2. Informational References

[BEAST] Thai, T. and J. Juliano, "Here Come The xor Ninjas", , May 2011, <https://www.researchgate.net/publication/266529975_Here_Come_The_Ninjas>.

[I-D.yeung-g-ikev2] Weis, B., Nir, Y., and V. Smyslov, "Group Key Management using IKEv2", draft-yeung-g-ikev2-11 (work in progress), March 2017.

Authors' Addresses

Daniel Migault (editor)
Ericsson
8400 boulevard Decarie
Montreal, QC H4P 2N2
Canada

Email: daniel.migault@ericsson.com

Tobias Guggemos (editor)
LMU Munich
Oettingenstr. 67
80538 Munich, Bavaria
Germany

Email: guggemos@mnm-team.org
URI: http://mnm-team.org/~guggemos

Yoav Nir
Dell EMC
9 Andrei Sakharov St
Haifa 3190500
Israel

Email: ynir.ietf@gmail.com