Internet Engineering Task Force (IETF)
Request for Comments: 5723
Category: Standards Track
ISSN: 2070-1721

Y. Sheffer
Check Point
H. Tschofenig
Nokia Siemens Networks

January 2010

Internet Key Exchange Protocol Version 2 (IKEv2) Session Resumption

Abstract

The Internet Key Exchange version 2 (IKEv2) protocol has a certain computational and communication overhead with respect to the number of round trips required and the cryptographic operations involved. In remote access situations, the Extensible Authentication Protocol (EAP) is used for authentication, which adds several more round trips and consequently latency.

To re-establish security associations (SAs) upon a failure recovery condition is time consuming especially when an IPsec peer (such as a VPN gateway) needs to re-establish a large number of SAs with various endpoints. A high number of concurrent sessions might cause additional problems for an IPsec peer during SA re-establishment.

In order to avoid the need to re-run the key exchange protocol from scratch, it would be useful to provide an efficient way to resume an IKE/IPsec session. This document proposes an extension to IKEv2 that allows a client to re-establish an IKE SA with a gateway in a highly efficient manner, utilizing a previously established IKE SA.

A client can reconnect to a gateway from which it was disconnected. The proposed approach encodes partial IKE state into an opaque ticket, which can be stored on the client or in a centralized store, and is later made available to the IKEv2 responder for re-authentication. We use the term ticket to refer to the opaque data that is created by the IKEv2 responder. This document does not specify the format of the ticket but examples are provided.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc5723.

Table of Contents

1. Introduction

The Internet Key Exchange version 2 (IKEv2) protocol has a certain computational and communication overhead with respect to the number of round trips required and the cryptographic operations involved. In particular, the Extensible Authentication Protocol (EAP) is used for authentication in remote access cases, which increases latency.

To re-establish security associations (SAs) upon a failure recovery condition is time-consuming, especially when an IPsec peer, such as a VPN gateway, needs to re-establish a large number of SAs with various endpoints. A high number of concurrent sessions might cause additional problems for an IPsec responder. Usability is also affected when the re-establishment of an IKE SA involves user interaction for re-authentication.

In many failure cases, it would be useful to provide an efficient way to resume an interrupted IKE/IPsec session. This document proposes an extension to IKEv2 that allows a client to re-establish an IKE SA with a gateway in a highly efficient manner, utilizing a previously established IKE SA.

The client (IKEv2 initiator) stores the state about the previous IKE SA locally. The gateway (IKEv2 responder) has two options for maintaining the IKEv2 state about the previous IKE SA:

- In the "ticket by reference" approach, the gateway stores the state locally, and gives the client a protected and opaque reference (e.g., an index to the gateway's table) that the gateway can later use to find the state. The client includes this opaque reference when it resumes the session.

- In the "ticket by value" approach, the gateway stores its state in a ticket (data structure) that is encrypted and integrity-protected by a key known only to the gateway. The ticket is passed to the client (who treats the ticket as an opaque string) and sent back to the gateway when the session is resumed. The gateway can then decrypt the ticket and recover the state.

Note that the client behaves identically in both cases, and in general does not know which approach the gateway is using. Since the ticket (or reference) is only interpreted by the same party that created it, this document does not specify the exact format for it. However, Appendix A contains examples for both "ticket by reference" and "ticket by value" formats.

This approach is similar to the one taken by Transport Layer Security (TLS) session resumption [RFC5077] with the required adaptations for IKEv2, e.g., to accommodate the two-phase protocol structure. We have borrowed heavily from that specification.

The proposed solution should additionally meet the following goals:

- Using only symmetric cryptography to minimize CPU consumption.
- Providing cryptographic agility.
- Having no negative impact on IKEv2 security features.

The following are non-goals of this solution:

- Failover from one gateway to another. This use case may be added in a future specification.
- Providing load balancing among gateways.
- Specifying how a client detects the need for resumption.

2. Terminology

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

This document uses terminology defined in [RFC4301] and [RFC4306]. In addition, this document uses the following term:

Ticket: An IKEv2 ticket is a data structure that contains all the necessary information that allows an IKEv2 responder to re-establish an IKEv2 security association.

In this document, we use the term "ticket" and thereby refer to an opaque data structure that may either contain IKEv2 state as described above or a reference pointing to such state.

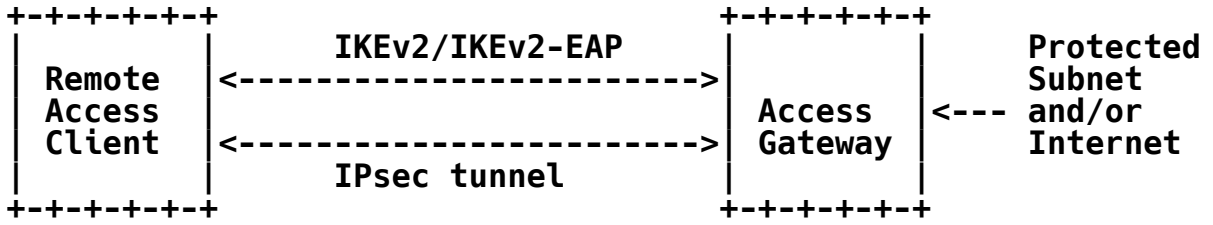
3. Usage Scenario

This specification envisions two usage scenarios for efficient IKEv2 and IPsec SA session re-establishment.

The first is similar to the use case specified in Section 1.1.3 of the IKEv2 specification [RFC4306], where the IPsec tunnel mode is used to establish a secure channel between a remote access client and a gateway; the traffic flow may be between the client and entities beyond the gateway. This scenario is further discussed below.

The second use case focuses on the usage of transport (or tunnel) mode to secure the communicate between two endpoints (e.g., two servers). The two endpoints have a client-server relationship with respect to a protocol that runs using the protections afforded by the IPsec SA.

(a)



(b)

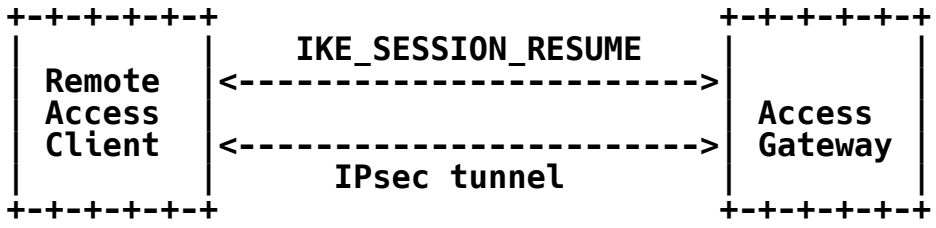


Figure 1: Resuming a Session with a Remote Access Gateway

In the first use case above, an end host (an entity with a host implementation of IPsec [RFC4301]) establishes a tunnel mode IPsec SA with a gateway in a remote network using IKEv2. The end host in this scenario is sometimes referred to as a remote access client. At a later stage, when a client needs to re-establish the IKEv2 session, it may choose to establish IPsec SAs using a full IKEv2 exchange or the IKE_SESSION_RESUME exchange (shown in Figure 1).

For either of the above use cases, there are multiple possible situations where the mechanism specified in this document could be useful. These include the following (note that this list is not meant to be exhaustive, and any particular deployment may not care about all of these):

- If a client temporarily loses network connectivity (and the IKE SA times out through the liveness test facility, a.k.a. "dead peer detection"), this mechanism could be used to re-establish the SA with less overhead (network, CPU, authentication infrastructure) and without requiring user interaction for authentication.

- If the connectivity problems affect a large number of clients (e.g., a large remote access VPN gateway), when the connectivity is restored, all the clients might reconnect almost simultaneously. This mechanism could be used to reduce the load spike for cryptographic operations and authentication infrastructure.

- Losing connectivity can also be predictable and planned; for example, putting a laptop to "stand-by" mode before traveling. This mechanism could be used to re-establish the SA when the laptop is switched back on (again, with less overhead and without requiring user interaction for authentication). However, such user-level "resumption" may often be disallowed by policy. Moreover, this document requires the client to destroy the ticket when the user explicitly "logs out" (Section 6.2).

4. Protocol Sequences

This section provides protocol details and contains the normative parts. This document defines two protocol exchanges, namely requesting a ticket, see Section 4.1, and presenting a ticket, see Section 4.3.

4.1. Requesting a Ticket

A client MAY request a ticket in the following exchanges:

- In an IKE_AUTH exchange, as shown in the example message exchange in Figure 2 below.

- In a CREATE_CHILD_SA exchange, when an IKE SA is rekeyed (and only when this exchange is initiated by the client).

- In an Informational exchange at any time, e.g., if the gateway previously replied with an N(TICKET_ACK) instead of providing a ticket, or when the ticket lifetime is about to expire, or following a gateway-initiated IKE rekey. All such Informational exchanges MUST be initiated by the client.

- While resuming an IKE session, i.e., in the IKE_AUTH exchange that follows an IKE_SESSION_RESUME exchange, see Section 4.3.3.

Normally, a client requests a ticket in the third message of an IKEv2 exchange (the first of IKE_AUTH). Figure 2 shows the message exchange for this typical case.

```
   Initiator                Responder
  -----------              -----------
 HDR, SAi1, KEi, Ni  -->

                     <--    HDR, SAr1, KEr, Nr [, CERTREQ]

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

Figure 2: Example Message Exchange for Requesting a Ticket

The notification payloads are described in Section 7. The above is an example, and IKEv2 allows a number of variants on these messages. Refer to [RFC4306] and [IKEV2-BIS] for more details on IKEv2.

When an IKEv2 responder receives a request for a ticket using the N(TICKET_REQUEST) payload, it MUST perform one of the following operations if it supports the extension defined in this document:

- it creates a ticket and returns it with the N(TICKET_LT_OPAQUE) payload in a subsequent message towards the IKEv2 initiator. This is shown in Figure 3.

- it returns an N(TICKET_NACK) payload, if it refuses to grant a ticket for some reason.

- it returns an N(TICKET_ACK), if it cannot grant a ticket immediately, e.g., due to packet size limitations. In this case, the client MAY request a ticket later using an Informational exchange, at any time during the lifetime of the IKE SA.

Regardless of this choice, there is no change to the behavior of the responder with respect to the IKE exchange, and the proper IKE response (e.g., an IKE_AUTH response or an error notification) MUST be sent.

4.2. Receiving a Ticket

The IKEv2 initiator receives the ticket and may accept it, provided the IKEv2 exchange was successful. The ticket may be used later with an IKEv2 responder that supports this extension. Figure 3 shows how the initiator receives the ticket.

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

Figure 3: Receiving a Ticket

When a multi-round-trip IKE_AUTH exchange is used, the N(TICKET_REQUEST) payload MUST be included in the first IKE_AUTH request, and N(TICKET_LT_OPAQUE) (or TICKET_NACK/TICKET_ACK) MUST only be returned in the final IKE_AUTH response.

When the client accepts the ticket, it stores it in its local storage for later use, along with the IKE SA to which the ticket refers. Since the ticket itself is opaque to the client, the local storage MUST also include all items marked as "from the ticket" in the table of Section 5.

4.3. Presenting a Ticket

When the client wishes to recover from an interrupted session, it presents the ticket to resume the session. This section describes the resumption process, consisting of some preparations, an IKE_SESSION_RESUME exchange, an IKE_AUTH exchange and finalization.

4.3.1. Prologue

It is up to the client's local policy to decide when the communication with the IKEv2 responder is seen as interrupted and the session resumption procedure is to be initiated.

A client MAY initiate a regular (non-ticket-based) IKEv2 exchange even if it is in possession of a valid, unexpired ticket. A client MUST NOT present a ticket when it knows that the ticket's lifetime has expired.

Tickets are intended for one-time use, i.e., a client MUST NOT reuse a ticket. A reused ticket SHOULD be rejected by a gateway. Note that a ticket is considered as used only when an IKE SA has been established successfully with it.

4.3.2. IKE_SESSION_RESUME Exchange

This document specifies a new IKEv2 exchange type called IKE_SESSION_RESUME whose value is 38. This exchange is equivalent to the IKE_SA_INIT exchange, and MUST be followed by an IKE_AUTH exchange. The client SHOULD NOT use this exchange type unless it knows that the gateway supports it (this condition is trivially true in the context of the current document, since the client always resumes into the same gateway that generated the ticket).

```
  Initiator                Responder
 -----------              -----------
 HDR, [N(COOKIE),] Ni, N(TICKET_OPAQUE) [,N+]   -->
```

Figure 4: IKEv2 Initiator Wishes to Resume an IKE SA

The exchange type in HDR is set to 'IKE_SESSION_RESUME'. The initiator sets the SPIi (Security Parameter Index, Initiator) value in the HDR to a new, unique value and the SPIr value is set to 0.

When the IKEv2 responder receives a ticket using the N(TICKET_OPAQUE) payload, it MUST perform one of the following steps if it supports the extension defined in this document:

- If it is willing to accept the ticket, it responds as shown in Figure 5.
- It responds with an unprotected N(TICKET_NACK) notification, if it rejects the ticket for any reason. In that case, the initiator should re-initiate a regular IKE exchange. One such case is when the responder receives a ticket for an IKE SA that has previously been terminated on the responder itself, which may indicate inconsistent state between the IKEv2 initiator and the responder. However, a responder is not required to maintain the state for terminated sessions.

```
  Initiator                Responder
 -----------              -----------
                <--  HDR, Nr [,N+]
```

Figure 5: IKEv2 Responder Accepts the Ticket

Again, the exchange type in HDR is set to 'IKE_SESSION_RESUME'. The responder copies the SPIi value from the request, and the SPIr value is set to a new, unique value.

Where not specified otherwise, the IKE_SESSION_RESUME exchange behaves exactly like the IKE_SA_INIT exchange. Specifically:

- The client MAY resume the IKE exchange from any IP address and port, regardless of its original address. The gateway MAY reject the resumed exchange if its policy depends on the client's address (although this rarely makes sense).
- The first message MAY be rejected in denial-of-service (DoS) situations, with the initiator instructed to send a cookie.
- Notifications normally associated with IKE_SA_INIT can be sent. In particular, NAT detection payloads.
- The client's NAT traversal status SHOULD be determined anew in IKE_SESSION_RESUME. If NAT is detected, the initiator switches to UDP encapsulation on port 4500, as per [RFC4306], Section 2.23. NAT status is explicitly not part of the session resumption state.
- The SPI values and Message ID fields behave similarly to IKE_SA_INIT.

Although the IKE SA is not fully valid until the completion of the IKE_AUTH exchange, the peers must create much of the SA state (Section 5) now. Specifically, the shared key values are required to protect the IKE_AUTH payloads. Their generation is described in Section 5.1.

4.3.3. IKE_AUTH Exchange

Following the IKE_SESSION_RESUME exchange, the client MUST initiate an IKE_AUTH exchange, which is largely as specified in [RFC4306]. This section lists the differences and constraints compared to the base document.

The value of the AUTH payload is derived in a manner similar to the usage of IKEv2 pre-shared secret authentication:

```
AUTH = prf(SK_px, <message octets>)
```

Each of the initiator and responder uses its own value for SK_px, namely SK_pi for the initiator and SK_pr for the responder. Both are taken from the newly generated IKE SA (Section 5.1).

The exact material to be signed is defined in Section 2.15 of [RFC4306].

The IDi value sent in the IKE_AUTH exchange MUST be identical to the value included in the ticket. A CERT payload MUST NOT be included in this exchange, and therefore a new IDr value cannot be negotiated (since it would not be authenticated). As a result, the IDr value sent (by the gateway, and optionally by the client) in this exchange MUST also be identical to the value included in the ticket.

When resuming a session, a client will typically request a new ticket immediately, so that it is able to resume the session again in the case of a second failure. The N(TICKET_REQUEST) and N(TICKET_LT_OPAQUE) notifications will be included in the IKE_AUTH exchange that follows the IKE_SESSION_RESUME exchange, with similar behavior to a ticket request during a regular IKE exchange, Section 4.1. The returned ticket (if any) will correspond to the IKE SA created per the rules described in Section 5.

4.3.4. Epilogue

Following the IKE_AUTH exchange, a new IKE SA is created by both parties, see Section 5, and a Child SA is derived, per Section 2.17 of [RFC4306].

When the responder receives a ticket for an IKE SA that is still active and if the responder accepts it (i.e., following successful completion of the IKE_AUTH exchange), the old SA SHOULD be silently deleted without sending a DELETE informational exchange. Consequently, all the dependent IPsec Child SAs are also deleted.

5. IKE and IPsec State after Resumption

During the resumption process, both peers create IKE and IPsec state for the resumed IKE SA. Although the SA is only completed following a successful IKE_AUTH exchange, many of its components are created earlier, notably the SA's crypto material (Section 5.1).

When a ticket is presented, the gateway needs to obtain the ticket state. In case a "ticket by reference" was provided by the client, the gateway needs to resolve the reference in order to obtain this state. In case the client has already provided a "ticket by value", the gateway can parse the ticket to obtain the state directly. In either case, the gateway needs to process the ticket state in order to restore the state of the old IKE SA, and the client retrieves the same state from its local store.

The following table describes the IKE and IPsec state of the peers after session resumption, and how it is related to their state before the IKE SA was interrupted. When the table mentions that a certain state item is taken "from the ticket", this should be construed as:

- The client retrieves this item from its local store.
- In the case of "ticket by value", the gateway encodes this information in the ticket.
- In the case of "ticket by reference", the gateway fetches this information from the ticket store.

State Item	After Resumption
IDi	From the ticket (but must also be exchanged in IKE_AUTH). See also Note 1.
IDr	From the ticket (but must also be exchanged in IKE_AUTH).
Authentication method (PKI, pre-shared secret, EAP, PKI-less EAP [EAP-AUTH] etc.)	From the ticket.
Certificates (when applicable)	From the ticket, see Note 2.
Local IP address/port, peer IP address/port	Selected by the client, see Note 3.
NAT detection status	From new exchange.
SPIs	From new exchange, see Note 4.
Which peer is the "original initiator"?	Determined by the initiator of IKE_SESSION_RESUME.
IKE SA sequence numbers (Message ID)	Reset to 0 in IKE_SESSION_RESUME, and subsequently incremented normally.
IKE SA algorithms (SAr)	From the ticket.

IKE SA keys (SK_*)	The old SK_d is obtained from the ticket and all keys are refreshed, see Section 5.1.
IKE SA window size	Reset to 1.
Child SAs (ESP/AH)	Created in new exchange, see Note 6.
Internal IP address	Not resumed, but see Note 5.
Other Configuration Payload information	Not resumed.
Peer Vendor IDs	Not resumed, resent in new exchange if required.
Peer supports MOBIKE [RFC4555]	From new exchange.
MOBIKE additional addresses	Not resumed, should be resent by client if necessary.
Time until re-authentication [RFC4478]	From new exchange (but ticket lifetime is bounded by this duration).
Peer supports redirects [RFC5685]	From new exchange.

Note 1: The authenticated peer identity used for policy lookups may not be the same as the IDi payload. This is possible when using certain EAP methods, see Section 3.5 of [RFC4718]. If these identities are indeed different, then the authenticated client identity MUST be included in the ticket. Note that the client may not have access to this value.

Note 2: Certificates don't need to be stored if the peer never uses them for anything after the IKE SA is up; however, if they are needed, e.g., if exposed to applications via IPsec APIs, they MUST be stored in the ticket.

Note 3: If the certificate has an iPAddress SubjectAltName, and the implementation requires it to match the peer's source IP address, the same check needs to be performed on session resumption and the required information saved locally or in the ticket.

Note 4: SPI values of the old SA MAY be stored in the ticket, to help the gateway locate corresponding old IKE state. These values MUST NOT be used for the resumed SA.

Note 5: The client can request the address it was using earlier, and if possible, the gateway SHOULD honor the request.

Note 6: Since information about Child SAs and configuration payloads is not resumed, IKEv2 features related to Child SA negotiation (such as IPCOMP_SUPPORTED, ESP_TFC_PADDING_NOT_SUPPORTED, ROHC-over-IPsec [ROHCoIPsec] and configuration) aren't usually affected by session resumption.

IKEv2 features that affect only the IKE_AUTH exchange (including HTTP_CERT_LOOKUP_SUPPORTED, multiple authentication exchanges [RFC4739], Elliptic Curve Digital Signature Algorithm (ECDSA) authentication [RFC4754], and the Online Certificate Status Protocol (OCSP) [RFC4806]) don't usually need any state in the IKE SA (after the IKE_AUTH exchanges are done), so resumption doesn't affect them.

New IKEv2 features that are not covered by Note 6 or by the previous paragraph should specify how they interact with session resumption.

5.1. Generating Cryptographic Material for the Resumed IKE SA

The cryptographic material is refreshed based on the ticket and the nonce values, Ni, and Nr, from the current exchange. A new SKEYSEED value is derived as follows:

```
SKEYSEED = prf(SK_d_old, "Resumption" | Ni | Nr)
```

where SK_d_old is taken from the ticket. The literal string is encoded as 10 ASCII characters, with no NULL terminator.

The keys are derived as follows, unchanged from IKEv2:

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

where SPIi, SPIr are the SPI values created in the new IKE exchange.

See [RFC4306] for the notation. "prf" is determined from the SA value in the ticket.

6. Ticket Handling

6.1. Ticket Content

When passing a "ticket by value" to the client, the ticket content MUST be integrity protected and encrypted.

A "ticket by reference" does not need to be encrypted, as it does not contain any sensitive material, such as keying material. However, access to the storage where that sensitive material is stored MUST be protected so that only authorized access is allowed. We note that such a ticket is analogous to the concept of 'stub', as defined in [SA-SYNC], or the concept of a Session ID from TLS.

Although not strictly required for cryptographic protection, it is RECOMMENDED to integrity-protect the "ticket by reference". Failing to do so could result in various security vulnerabilities on the gateway side, depending on the format of the reference. Potential vulnerabilities include access by the gateway to unintended URLs (similar to cross-site scripting) or SQL injection.

When the state is passed by value, the ticket MUST encode all state information marked "from the ticket" in the table on Section 5. The same state MUST be stored in the ticket store, in the case of "ticket by reference".

A "ticket by value" MUST include a protected expiration time, which is an absolute time value and SHOULD correspond to the value included in the TICKET_LT_OPAQUE payload.

The "ticket by value" MUST additionally include a key identity field, so that keys for ticket encryption and authentication can be changed, and when necessary, algorithms can be replaced.

6.2. Ticket Identity and Lifecycle

Each ticket is associated with a single IKE SA. In particular, when an IKE SA is deleted by the client or the gateway, the client MUST delete its stored ticket. Similarly, when credentials associated with the IKE SA are invalidated (e.g., when a user logs out), the ticket MUST be deleted. When the IKE SA is rekeyed, the ticket is invalidated, and the client SHOULD request a new ticket. When a client does not follow these rules, it might present an invalid ticket to the gateway. See Section 9.8 for more about this issue.

The lifetime of the ticket sent by the gateway SHOULD be the minimum of the IKE SA lifetime (per the gateway's local policy) and its re-authentication time, according to [RFC4478]. Even if neither of these are enforced by the gateway, a finite lifetime MUST be specified for the ticket.

The key that is used to protect the ticket MUST have a lifetime that is significantly longer than the lifetime of an IKE SA.

In normal operation, the client will request a ticket when establishing the initial IKE SA, and then every time the SA is rekeyed or re-established because of re-authentication.

7. IKE Notifications

This document defines a number of notifications. The following Notify Message types have been assigned by IANA.

Notification Name	Value	Data
TICKET_LT_OPAQUE	16409	See Section 7.1
TICKET_REQUEST	16410	None
TICKET_ACK	16411	None
TICKET_NACK	16412	None
TICKET_OPAQUE	16413	See Section 7.2

For all these notifications, the Protocol ID and the SPI Size fields MUST both be sent as 0.

7.1. TICKET_LT_OPAQUE Notify Payload

The data for the TICKET_LT_OPAQUE Notify payload consists of the Notify message header, a Lifetime field and the ticket itself. The four octet Lifetime field contains a relative time value, the number of seconds until the ticket expires (encoded as an unsigned integer, in network byte order).

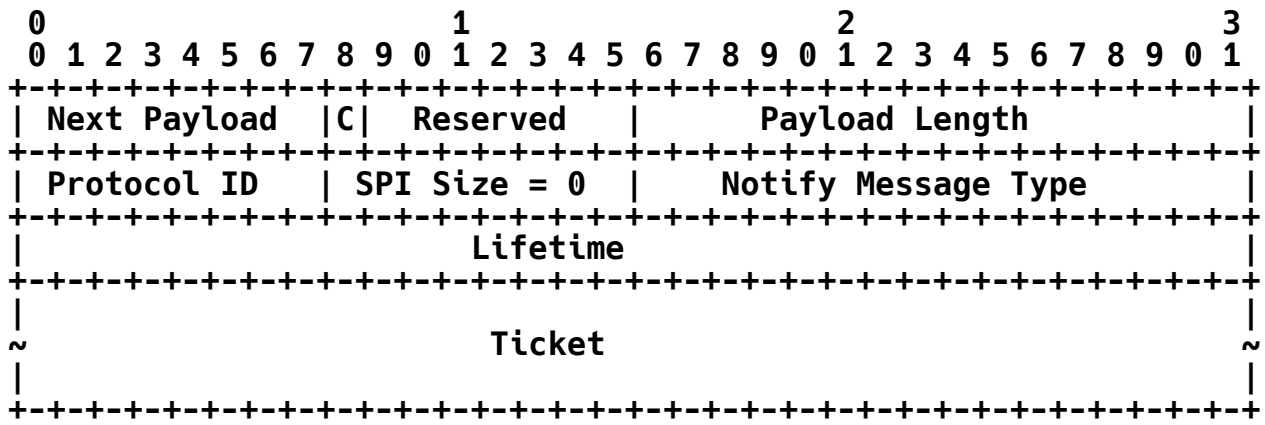


Figure 6: TICKET_LT_OPAQUE Notify Payload

7.2. TICKET_OPAQUE Notify Payload

The data for the TICKET_OPAQUE Notify payload consists of the Notify message header, and the ticket itself. Unlike the TICKET_LT_OPAQUE payload, no lifetime value is included in the TICKET_OPAQUE Notify payload.

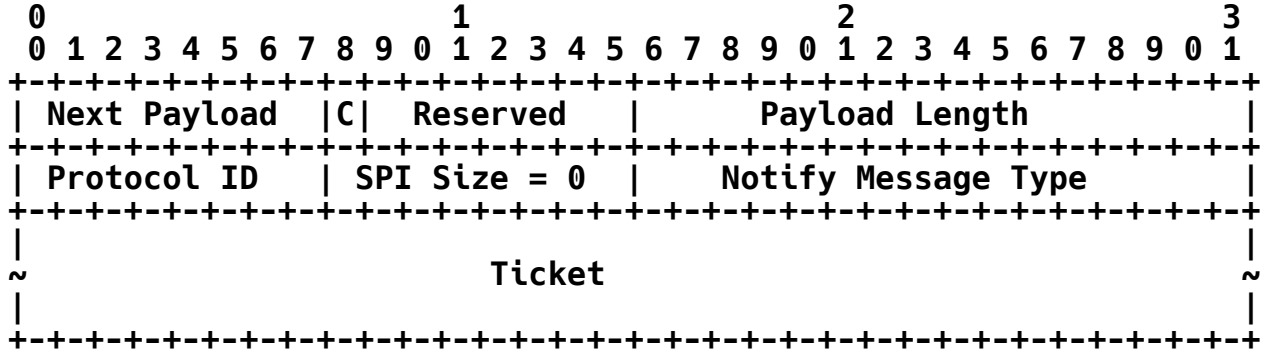


Figure 7: TICKET_OPAQUE Notify Payload

8. IANA Considerations

Section 4.3.2 defines a new IKEv2 exchange type, IKE_SESSION_RESUME, whose value has been allocated from the "IKEv2 Exchange Types" registry.

Section 7 defines several new IKEv2 notifications whose Message Type values have been allocated from the "IKEv2 Notify Message Types - Status Types" registry.

9. Security Considerations

This section addresses security issues related to the usage of a ticket.

9.1. Stolen Tickets

A man in the middle may try to eavesdrop on an exchange to obtain a "ticket by value" and use it to establish a session with the IKEv2 responder. Since all exchanges where the client obtains a ticket are encrypted, this is only possible by listening in on a client's use of the ticket to resume a session. However, since the ticket's contents are encrypted and the attacker does not know the corresponding secret key, a stolen ticket cannot be used by an attacker to successfully resume a session. An IKEv2 responder MUST use strong encryption and integrity protection of the ticket to prevent an attacker from obtaining the ticket's contents, e.g., by using a brute force attack.

A "ticket by reference" does not need to be encrypted. When an adversary is able to eavesdrop on a resumption attempt, as described in the previous paragraph, then the "ticket by reference" may be obtained. A "ticket by reference" cannot be used by an attacker to successfully resume a session, for the same reasons as for a "ticket by value", namely because the attacker would not be able to prove, during IKE_AUTH, its knowledge of the secret part of the IKE state embedded in the ticket. Moreover, the adversary MUST NOT be able to resolve the ticket via the reference, i.e., access control MUST be enforced to ensure disclosure only to authorized entities.

9.2. Forged Tickets

A malicious user could forge or alter a "ticket by value" in order to resume a session, to extend its lifetime, to impersonate as another user, or to gain additional privileges. This attack is not possible if the content of the "ticket by value" is protected using a strong integrity protection algorithm.

In the case of a "ticket by reference" an adversary may attempt to construct a fake "ticket by reference" to point to state information stored by the IKEv2 responder. This attack will fail because the adversary is not in possession of the keying material associated with the IKEv2 SA. As noted in Section 6.1, it is often useful to integrity-protect the "ticket by reference", too.

9.3. Denial-of-Service Attacks

An adversary could generate and send a large number of "tickets by value" to a gateway for verification. Such an attack could burden the gateway's CPU, and/or exhaust its memory with half-open IKE state. To minimize the possibility of such denial of service, ticket verification should be lightweight (e.g., using efficient symmetric key cryptographic algorithms).

When an adversary chooses to send a large number of "tickets by reference" then this may lead to an amplification attack as the IKEv2 responder is forced to resolve the reference to a ticket in order to determine that the adversary is not in possession of the keying material corresponding to the stored state or that the reference is void. To minimize this attack, the protocol to resolve the reference should be as lightweight as possible and should not generate a large number of messages.

Note also that the regular IKEv2 cookie mechanism can be used to handle state-overflow DoS situations.

9.4. Detecting the Need for Resumption

Detecting when an old IKE SA is no longer usable and needs to be resumed is out of scope of the current document. However, clients are warned against implementing a more liberal policy than that used to detect failed IKE SAs (Section 2.4 of RFC 4306). In particular, untrusted messages MUST NOT be relied upon to make this decision.

9.5. Key Management for "Tickets by Value"

A full description of the management of the keys used to protect the "ticket by value" is beyond the scope of this document. A list of RECOMMENDED practices is given below.

- The keys should be generated securely following the randomness recommendations in [RFC4086].
- The keys and cryptographic protection algorithms should be at least 128 bits in strength.
- The keys should not be used for any other purpose than generating and verifying tickets.
- The keys should be changed regularly.
- The keys should be changed if the ticket format or cryptographic protection algorithms change.

9.6. Ticket Lifetime

An IKEv2 responder controls the validity period of the state information by attaching a lifetime to a ticket. The chosen lifetime is based on the operational and security requirements of the environment in which this IKEv2 extension is deployed. The responder provides information about the ticket lifetime to the IKEv2 initiator, allowing it to manage its tickets.

9.7. Tickets and Identity

A ticket is associated with a certain identity, and MUST be managed securely on the client side. Section 6.2 requires that a ticket be deleted when the credentials associated with the ticket's identity are no longer valid, e.g., when a user whose credentials were used to create the SA logs out.

9.8. Ticket Revocation

A misbehaving client could present a ticket in its possession to the gateway resulting in session resumption, even though the IKE SA associated with this ticket had previously been deleted. This is disallowed by Section 6.2. This issue is unique to "ticket by value" cases, since a "ticket by reference" will have been deleted from the ticket store.

To avoid this issue for "ticket by value", an Invalid Ticket List (ITL) may be maintained by the gateway, see [TOKENS]. This can be a simple blacklist of revoked tickets. Alternatively, [TOKENS] suggests to use Bloom Filters [Bloom70] to maintain the list in constant space. Management of such lists is outside the scope of the current document. Note that a policy that requires tickets to have shorter lifetimes (e.g., 1 hour) significantly mitigates this issue.

9.9. Ticket by Value Format

The ticket's format is not defined by this document, since this is not required for interoperability. However, great care must be taken when defining a ticket format such that the requirements outlined in Section 6.1 are met. The "ticket by value" MUST have its integrity and confidentiality protected with strong cryptographic techniques to prevent a breach in the security of the system.

9.10. Identity Privacy, Anonymity, and Unlinkability

Since opaque state information is passed around between the IKEv2 initiator and the IKEv2 responder it is important that leakage of information, such as the identities of an IKEv2 initiator and a responder, MUST be avoided.

When an IKEv2 initiator presents a ticket as part of the IKE_SESSION_RESUME exchange, confidentiality is not provided for the exchange. There is thereby the possibility for an on-path adversary to observe multiple exchange handshakes where the same state information is used and therefore to conclude that they belong to the same communication endpoints.

This document therefore requires that the ticket be presented to the IKEv2 responder only once; under normal circumstances (e.g., no active attacker), there should be no multiple use of the same ticket.

We are not aware of additional security issues associated with ticket reuse: the protocol guarantees freshness of the generated crypto material even in such cases. As noted in Section 4.3.1, the gateway SHOULD prevent multiple uses of the same ticket. But this is only an extra precaution, to ensure that clients do not implement reuse. In other words, the gateway is not expected to cache old tickets for extended periods of time.

10. Acknowledgements

We would like to thank Paul Hoffman, Pasi Eronen, Florian Tegeler, Stephen Kent, Sean Shen, Xiaoming Fu, Stjepan Gros, Dan Harkins, Russ Housely, Yoav Nir, Peny Yang, Sean Turner, and Tero Kivinen for their comments. We would like to particularly thank Florian Tegeler and Stjepan Gros for their implementation efforts and Florian Tegeler for a formal verification using the Casper tool set.

We would furthermore like to thank the authors of [SA-SYNC] (Yan Xu, Peny Yang, Yuanchen Ma, Hui Deng, and Ke Xu) for their input on the stub concept.

We would like to thank Hui Deng, Tero Kivinen, Peny Yang, Ahmad Muhanna, and Stephen Kent for their feedback regarding the "ticket by reference" concept.

Vidya Narayanan and Lakshminath Dondeti coauthored several past versions of this document, and we acknowledge their significant contribution.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

11.2. Informative References

[Bloom70] Bloom, B., "Space/time trade-offs in hash coding with allowable errors", Comm. ACM 13(7):422-6, July 1970.

[EAP-AUTH] Eronen, P., Tschofenig, H., and Y. Sheffer, "An Extension for EAP-Only Authentication in IKEv2", Work in Progress, October 2009.

[IKEV2-BIS] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol: IKEv2", Work in Progress, October 2009.

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4478] Nir, Y., "Repeated Authentication in Internet Key Exchange (IKEv2) Protocol", RFC 4478, April 2006.

[RFC4555] Eronen, P., "IKEv2 Mobility and Multihoming Protocol (MOBIKE)", RFC 4555, June 2006.

[RFC4718] Eronen, P. and P. Hoffman, "IKEv2 Clarifications and Implementation Guidelines", RFC 4718, October 2006.

[RFC4739] Eronen, P. and J. Korhonen, "Multiple Authentication Exchanges in the Internet Key Exchange (IKEv2) Protocol", RFC 4739, November 2006.

[RFC4754] Fu, D. and J. Solinas, "IKE and IKEv2 Authentication Using the Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 4754, January 2007.

[RFC4806] Myers, M. and H. Tschofenig, "Online Certificate Status Protocol (OCSP) Extensions to IKEv2", RFC 4806, February 2007.

[RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, January 2008.

[RFC5685] Devarapalli, V. and K. Weniger, "Redirect Mechanism for the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5685, November 2009.

[ROHCoIPsec] Ertekin, E., Christou, C., Jasani, R., Kivinen, T., and C. Bormann, "IKEv2 Extensions to Support Robust Header Compression over IPsec (ROHCoIPsec)", Work in Progress, December 2009.

[SA-SYNC] Xu, Y., Yang, P., Ma, Y., Deng, H., and H. Deng, "IKEv2 SA Synchronization for session resumption", Work in Progress, October 2008.

[TOKENS] Rescorla, E., "How to Implement Secure (Mostly) Stateless Tokens", Work in Progress, March 2007.

Appendix A. Ticket Format

This document does not specify a particular ticket format nor even the suggested contents of a ticket: both are entirely up to the implementer. The formats described in the following sub-sections are provided as useful examples, and implementers are free to adopt them as-is or change them in any way necessary.

A.1. Example "Ticket by Value" Format

```
  struct {
      [authenticated] struct {
          octet format_version;    // 1 for this version of the protocol
          octet reserved[3];       // sent as 0, ignored by receiver.
          octet key_id[8];         // arbitrary byte string
          opaque IV[0..255];       // actual length (possibly 0) depends
                                   // on the encryption algorithm

          [encrypted] struct {
              opaque IDi, IDr;     // the full payloads
              octet SPIi[8], SPIr[8];
              opaque SA;           // the full SAr payload
              octet SK_d[0..255];  // actual length depends on SA value
              enum ... authentication_method;
              int32 expiration;    // an absolute time value, seconds
                                   // since Jan. 1, 1970
          } ikev2_state;
      } protected_part;
      opaque MAC[0..255];          // the length (possibly 0) depends
                                   // on the integrity algorithm
  } ticket;
```

Note that the key defined by "key_id" determines the encryption and authentication algorithms used for this ticket. Those algorithms are unrelated to the transforms defined by the SA payload.

The reader is referred to [TOKENS] that recommends a similar (but not identical) ticket format, and discusses related security considerations in depth.

A.2. Example "Ticket by Reference" Format

For implementations that prefer to pass a reference to IKE state in the ticket, rather than the state itself, we suggest the following format:

```
  struct {
        [authenticated] struct {
            octet format_version;  // 1 for this version of the protocol
            octet reserved[3];     // sent as 0, ignored by receiver.
            octet key_id[8];       // arbitrary byte string

            struct {
                opaque state_ref;  // reference to IKE state
                int32 expiration;  // an absolute time value, seconds
                                   // since Jan. 1, 1970
            } ikev2_state_ref;
        } protected_part;
        opaque MAC[0..255];        // the length depends
                                   // on the integrity algorithm
  } ticket;
```

Authors' Addresses

Yaron Sheffer
Check Point Software Technologies Ltd.
5 Hasolelim St.
Tel Aviv 67897
Israel

EMail: yaronf@checkpoint.com

Hannes Tschofenig
Nokia Siemens Networks
Linnoitustie 6
Espoo 02600
Finland

Phone: +358 (50) 4871445
EMail: Hannes.Tschofenig@gmx.net
URI: http://www.tschofenig.priv.at