Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 1, 2021

V. Smyslov
ELVIS-PLUS

October 28, 2020

Revised Cookie Processing in the IKEv2 Protocol
draft-smyslov-ipsecme-ikev2-cookie-revised-00

Abstract

This document defines a revised processing of cookies in the Internet Key Exchange protocol Version 2 (IKEv2). It is intended to solve a problem in IKEv2 when due to packets loss and reordering peers may erroneously fail to authenticate each other when cookies are used in the initial IKEv2 exchange.

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 1, 2021.

Table of Contents

1. Introduction

The Internet Key Exchange protocol Version 2 (IKEv2) described in [RFC7296] includes mechanism to defend against DoS attacks. The mechanism is based on cookie which a responder can request an initiator to return in a subsequent request. This allows the responder avoid creating state until it is sure that the initiator's IP address is not spoofed. The cookie mechanism is optional and it is up to the responder whether to use it or not.

When cookie mechanism is used in networks with high probability of packets loss and reordering, it is possible that peers end up with different views on whether cookies were used or not or which content the used cookie had. Since cookie, if used, is a part of an IKEv2 message that is included into calculation of authentication data by both peers, the different views leads to the situation when peers erroneously fail to authenticate each other.

This specification revises processing of cookies in IKEv2 in such a way that peers supporting it exclude cookies from data to be authenticated. This allows them to complete authentication even in the situation described above.

2. Terminology and Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and

"OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Using Cookie in IKEv2

Section 2.6 of [RFC7296] specifies that when a responder detects a large number of half-open IKE SAs, it SHOULD reply to an IKE_SA_INIT request with a response containing the COOKIE notification and If an IKE_SA_INIT response includes the COOKIE notification, the initiator MUST then retry the IKE_SA_INIT request, and include the COOKIE notification containing the received data as the very first payload in it, retaining all other payloads intact. This is illustrated in the Figure 1.

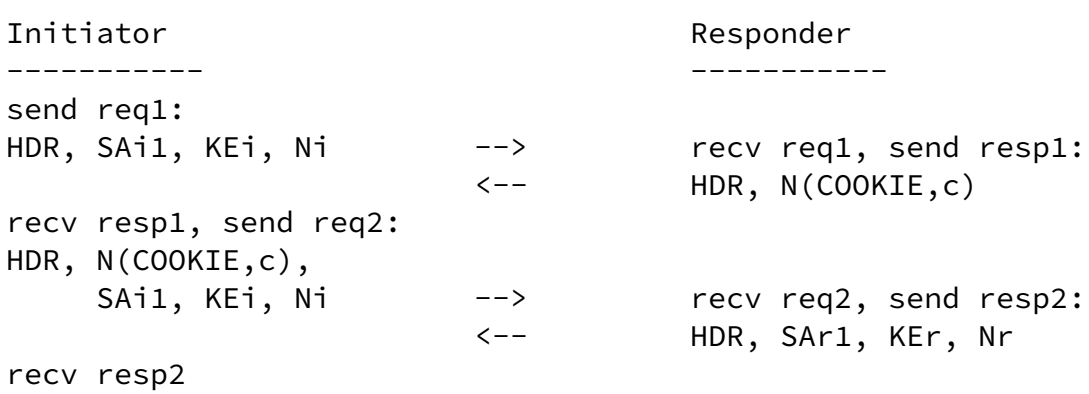

Figure 1

Note, that the responder saves no state when it sends message resp1. This is achieved by the way cookies are generated. A good way to generate a cookie is described in Section 2.6 of [RFC7296]:

Cookie = <VersionIDofSecret> | Hash(Ni | IPi | SPIi | <secret>)

where <secret> is a randomly generated secret known only to the responder and periodically changed. [RFC7296] advises the responder to change the value of <secret> frequently, especially if under attack.

Later in the IKE_AUTH exchange the IKE_SA_INIT messages are

authenticated by including their content intact into the data that is signed (or MAC'ed) using peers' credentials (see Section 2.15 of [RFC7296] for details).

4. Problem Description

To successfully complete authentication it is important that both peers use the same content of the IKE_SA_INIT messages when calculating authentication data. However, when cookie is employed, the IKE_SA_INIT request is sent at least twice with different content. Section 2.15. of [RFC7296] states that if the first message of the IKE_SA_INIT exchange is sent multiple times with different

content (such as with a cookie), it is the latest version of the message that is used for authentication. However, in situations when network packets can be lost and reordered peers may end up with different views on what is "the latest version of the message". Two examples of such situations are shown below.

Consider a situation when at the time the initiator starts creating IKE SA by sending req1 message the responder thinks it's under attack and responds with a resp2 message containing cookie request. However, this message is delayed in the network.

```
Initiator                               Responder
-----------                             -----------
send req1:
HDR, SAi1, KEi, Ni          -->         recv req1, send resp1:
 (delayed)                  <--         HDR, N(COOKIE,c)
```

Since the initiator hasn't received any response, it retransmits its initial request message req1 after some time. During this time the situation on the responder has changed and it doesn't think it's under attack anymore, so it responds with resp2 message and considers the IKE_SA_INIT exchange completed. This message is also delayed in the network.

```
Initiator                               Responder
-----------                             -----------
re-send req1:
HDR, SAi1, KEi, Ni          -->         recv req1, send resp2:
 (delayed)                  <--         HDR, SAr1, KEr, Nr
```

After some time the initiator eventually receives the first initiator's response resp1, which contains cookie request. It is the first response the initiator receives, so it re-sends the request adding the received cookie into it (req2). However, this message is lost and never reaches the responder. Shortly after sending req2 the initiator receives resp2 message.

```
Initiator                               Responder
-----------                             -----------
recv resp1, send req2:
HDR, N(COOKIE,c),
     SAi1, KEi, Ni          -->          (lost)
recv resp2
```

At this point both peers have completed the IKE_SA_INIT exchange and the KE_AUTH exchange is ready to start. However, the peers have different opinions on what the latest IKE_SA_INIT request message was - the initiator thinks it was req2, while the responder thinks it was

req1. As a result - the authentication in the IKE_AUTH exchange will fail.

Let's consider another possible sequence, that leads to the same result. As with the previous example the initiator starts creating IKE SA by sending req1 message. The responder thinks it's under attack and responds with a resp2 message containing cookie request with cookie c1. However, this message is delayed in the network.

```
Initiator                               Responder
-----------                             -----------
send req1:
HDR, SAi1, KEi, Ni          -->         recv req1, send resp1:
 (delayed)                  <--         HDR, N(COOKIE, c1)
```

As with the first example, the initiator hasn't received any response and retransmits its initial request message req1 after some time. It happens that within this time the value of <secret> has changed (note, that [RFC7296] advises to do it frequently, especially when under attack). Since the responder cannot verify cookie c1 and it still thinks it is under attack, it acts as if req1 contains no cookie and sends back resp2 message also containing cookie request,

with a new cookie c2 (that was calculated using the same input data and the new value of <secret>).

```
Initiator                               Responder
-----------                             -----------
re-send req1:
HDR, SAi1, KEi, Ni          -->         recv req1, send resp2:
                            <--         HDR, N(COOKIE, c2)
```

This message is not delayed, it reaches the initiator and the initiator sends a new request req2 containing cookie c2. The responder receives this message and successfully verifies the cookie using its current value of <secret>. Since the cookie verification is successfull, the responder sends back resp3 message and considers the IKE_SA_INIT exchange completed. However, resp3 message is delayed.

```
Initiator                               Responder
-----------                             -----------
recv resp2, send req2:
HDR, N(COOKIE, c2),
     SAi1, KEi, Ni          -->         recv req2, send resp3:
 (delayed)                  <--         HDR, SAr1, KEr, Nr
```

After some time the initiator eventually receives resp1 message, which was delayed. This message contains another value of cookie -

c1. Since this message was received later than resp2 message, the initiator thinks the value c1 is fresher than c2 and sends a new request message req3 now with c1 cookie. This message is lost in the network and never reaches the responder. Shortly after sending req3 the initiator receives resp3 message.

```
Initiator                               Responder
-----------                             -----------
recv resp1, send req3:
HDR, N(COOKIE, c1),
     SAi1, KEi, Ni          -->          (lost)
recv resp3
```

As with the first example, at this point both peers have completed the IKE_SA_INIT exchange and are ready for the KE_AUTH exchange.

However, the peers again have different opinions on what the latest IKE_SA_INIT request message was - the initiator thinks it was req3, while the responder thinks it was req2. As a result - the authentication in the IKE_AUTH exchange will fail as with the previous example.

The root of this problem is that the initial request can be re-sent several times with different content depending on the responder's current state, which can change over time. Note, that this situation is generally not possible with the INVALID_KE_PAYLOAD notification, even that in this case the request is also sent several times. This is because the responder will always either require changing Key Exchange method or not, so it is not possible that eventually peers end up with different opinions on what Key Exchange method was negotiated in the IKE_SA_INIT exchange.

5. Revised Cookie Processing

This specification proposes to solve the problem by excluding cookie content from data to be authenticated. The rationale for this is that cookie must be verified by the responder independently at the time it is received in the IKE_SA_INIT request, so there is no need to authenticate it.

5.1. Negotiation of Revised Cookie Processing

For the purpose of using revised cookie processing a new Status Type notify REVISED_COOKIE is defined. Its Notify Message Type is <TBA by IANA>, Protocol ID and SPI Size are both set to 0. The responder includes an empty REVISED_COOKIE notification whenever it sends a response containing COOKIE notification. If the initiator doesn't support this extension it will ignore this notification and continues as described in [RFC7296]. In case the initiator supports revised

cookie processing it will re-send its initial request including the received cookie, but placing the cookie data into the REVISED_COOKIE notification instead of COOKIE notification.

```
Initiator                               Responder
-----------                             -----------
send req1:
HDR, SAi1, KEi, Ni          -->         recv req1, send resp1:
```

```
                                        HDR, N(COOKIE,c),
                            <--              N(REVISED_COOKIE)
   recv resp1, send req2:
   HDR, N(REVISED_COOKIE,c),
        SAi1, KEi, Ni       -->         recv req2, send resp2:
                            <--         HDR, SAr1, KEr, Nr
   recv resp2
```

Figure 2

5.2. Processing of REVISED_COOKIE Notification

If the responder has sent the REVISED_COOKIE notification in the message requesting cookie it should be prepared to receive the re-sent IKE_SA_INIT request with the REVISED_COOKIE notification containing the cookie and with no COOKIE notification. Processing of the REVISED_COOKIE notification by the responder MUST be identical to the processing of COOKIE notification which is described in Sections 2.6 and 2.7 of [RFC7296].

5.3. Changes in AUTH Payload Calculation

In the subsequent IKE_AUTH exchange peers authenticate each other by signing (or MAC'ing) blobs of data. These blobs are defined in Section 2.15 of [RFC7296]. In particular, initiator's blob is defined as follows:

```
InitiatorSignedOctets = RealMessage1 | NonceRData | MACedIDForI
GenIKEHDR = [ four octets 0 if using port 4500 ] | RealIKEHDR
RealIKEHDR =  SPIi | SPIr |  . . . | Length
RealMessage1 = RealIKEHDR | RestOfMessage1
NonceRPayload = PayloadHeader | NonceRData
InitiatorIDPayload = PayloadHeader | RestOfInitIDPayload
RestOfInitIDPayload = IDType | RESERVED | InitIDData
MACedIDForI = prf(SK_pi, RestOfInitIDPayload)
```

Figure 3

In Figure 3 RealMessage1 is the latest version of the IKE_SA_INIT request message.

If the very first payload in RealMessage1 is the REVISED_COOKIE

notify, then InitiatorSignedOctets are computed as shown in Figure 4. In particular:

1. The content of the REVISED_COOKIE notify payload is eliminated from the message;

2. The Next Payload field in the IKE Header is set to the value of the Next Payload field in the header of eliminated payload;

3. The length of the eliminated payload (indicated in the Length field in its header) is subtracted from the Length field in the IKE Header.

```
InitiatorSignedOctets = PseudoMessage1 | NonceRData | MACedIDForI
RealMessage1 = RealIKEHDR | NotifyREVISED_COOKIE | RestOfMessage1
NotifyREVISED_COOKIE = NextPld | 0 | PldLength | RestOfNotifyCOOKIE
GenIKEHDR = [ four octets 0 if using port 4500 ] | RealIKEHDR
RealIKEHDR = SPIi | SPIr |  HdrNextPld | . . . | MsgLength
PseudoKEHDR =  SPIi | SPIr |  NewHdrNextPld | . . . | NewMsgLength
NewHdrNextPld = NextPld
NewMsgLength = MsgLength - PldLength
PseudoMessage1 = PseudoIKEHDR | RestOfMessage1
NonceRPayload = PayloadHeader | NonceRData
InitiatorIDPayload = PayloadHeader | RestOfInitIDPayload
RestOfInitIDPayload = IDType | RESERVED | InitIDData
MACedIDForI = prf(SK_pi, RestOfInitIDPayload)
```

Figure 4

In brief, if RealMessage1 doesn't contain the REVISED_COOKIE notification then it is used in the authentication as is (Figure 3). Otherwise a new pseudo message PseudoMessage1 is used which is constructed from RealMessage1 as if it doesn't contain the REVISED_COOKIE notification (Figure 4>).

This modification excludes Notify payload containing cookie from the input to the AUTH payload calculation, thus solving the problem described in Section 4.

6. Security Considerations

This extension modifies the way IKE initiator is authenticated to the IKE responder. In particular, the cookie, created by the responder and returned by the initiator in the IKE_SA_INIT request is excluded from the data to be authenticated. IKEv2 specification requires that cookie (if present in the request) be verified by the responder at the early stage of the IKE_SA_INIT request message processing. If

this verification fails, then the responder must act as if no cookie were present (see Section 2.6 of [RFC7296]), which in most cases results in requesting a new cookie. An adversary that is able to modify cookie content (or remove it from the request) will get no new advantages if this extension is used: either the responder requests a new cookie, or it doesn't care about the cookie at the moment and the IKE_SA_INIT exchange succeeded with invalid cookie. In the later case if revised cookie processing is used the subsequent IKE_AUTH exchange will also succeed and IKE SA will be created, which is different from the current situation, when authentication will fail in the IKE_AUTH if cookie is changed by the attacker.

Excluding cookie from the data to be authenticated doesn't degrade security properties of IKEv2, because the content of the cookie is generated by the responder and must be verified by the responder well before the authentication takes place. The initiator doesn't participate in generation of cookie, it only returns it back as a blob.

Compared to the current processing of cookie the difference caused by the revised processing in a situation when an attacker changes cookie in the IKE_SA_INIT request is that IKE SA will still be created (provided no other obstacles exists), but only if the responder at the moment doesn't care about validity of the received cookie (it means that it is not under attack).

7. IANA Considerations

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

<TBA>	REVISED_COOKIE

8. Acknowledgements

Author is grateful to Tero Kivinen and Wei Pan for sharing their thoughts about this problem and potential ways to solve it.

9. 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>.

[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>.

[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>.

Author's Address

Valery Smyslov
ELVIS-PLUS
PO Box 81
Moscow (Zelenograd) 124460
RU

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