Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 14, 2016

R. Van Rein
ARPA2.net

March 13, 2016

Kerberos in the Extensible Authentication Protocol (EAP-Kerberos) draft-vanrein-eap-kerberos-00

Abstract

This specification permits Kerberos authentication as part of the EAP. To support identity bootstrapping during network logon, the preceding acquisition of tickets may also be passed through this EAP mechanism.

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 September 14, 2016.

Copyright Notice

Table of Contents

1. Introduction

EAP, or the Extensible Authentication Protocol, is a pluggable mechanism for the control of network or service access by users. It is usually communicated with users during a login phase of PPP or 802.1x, and passed on to backend servers in a RADIUS attribute [Section 5.1 of [RFC2865]] or Diameter AVP [Section 8.14 of [RFC6733]] set aside for EAP.

Although often used with password-based policies, EAP is equally suited for more advanced cryptographic approaches. What this specification adds, is a facility for Kerberos authentication, without resorting to the use of a KDC as a mere password oracle for PAP. To make this possible, the Application Protocol (AP) exchange of Kerberos is facilitated over EAP.

Since EAP is so often used to bootstrap online access, as is the case with PPP and 802.1x use cases, additional care is needed for procuring Kerberos tickets to use in the AP exchange. To facilitate this, the EAP profile presented herein can also be used to pass Authentication Service (AS) and Ticket Granting Service (TGS) exchanges with a Kerberos-supportive backend.

For users, the applications of EAP are integrated with a Single Sign-On (SSO) facility, making their online experience more seemless

because no separate password entry is required, nor is it required to resort to fixed secrets installed on their systems.

The following Security Claims [Section 7.2 of [RFC3748]] are made for EAP-Kerberos:

Auth. mechanism:	Kerberos
Ciphersuite negotiation:	Yes
Mutual authentication:	Yes
Integrity protection:	No (not for EAP header fields)
Replay protection:	Yes
Confidentiality:	No, not always and not completely
Key derivation:	Yes
Key strength:	Usually 128 or 256 bit
Dictionary attack prot.:	No
Fast reconnect:	Yes
Crypt. binding:	Yes, as implied by Kerberos
Session independence:	Yes
Fragmentation:	Yes
Channel binding:	No

2. Inner and Outer Identities

It is common for EAP to be encapsulated in a context that communicates an identity, independently of what the EAP does with it. This identity is sometimes referred to as the "outer" identity, to contrast it with an "inner" identity negotiated within the EAP transport. As an example, when EAP is transported in RADIUS or Diameter, there is commonly a User-Name attribute at the same level as EAP; this would be the outer identity.

Not all methods distinguish between inner and outer identities and under such methods the outer identity often holds a user name to be authenticated. This is not strictly required in an EAP context; the only thing needed is a decision from the backend to Accept or Reject a peer; any data exchanged works towards that end result.

Under EAP-Kerberos, the client PrincipalName and Realm may be considered the inner identity, and indeed is this the identity that is validated by the protocol flow. It is vital to understand that the outer identity is in no way to be considered reliable as a user identity for the peer; this is because it falls outside the protected context of EAP-Kerberos.

The outer identity serves a different goal, namely as a locator for the backend authorisation server. To the peer, this is a backend that is supportive of its access, and to the authenticator this is a basis for locating and/or constraining the backend server as an

authoritative source of information about the peer and its rights of access to the EAP-protected resource.

3. Transporting Kerberos Frames over EAP

The EAP-Kerberos protocol carries an ordered flow of Kerberos frames, communicated in round trips. This facilitates a Kerberos query/response sequence initiated by the peer, and leading to eventual Succes or Failure to authenticate the peer. This section details what Kerberos frames are, and how an EAP message flow implements them.

Whenever a fatal error is detecting during the processing of EAP-Kerberos, this MUST lead to rejection of the authentication attempt, possibly after delivery of a final KRB-ERROR frame. Any conflict with the specifications is considered a fatal error.

3.1. EAP-Kerberos as a Flow of Kerberos Frames

EAP-Kerberos implements a flow of Kerberos frames. Each Kerberos frame is a single data structure, recognised by an [APPLICATION n] tag. The exchange between the authenticating peer and backend authentication server consists of a request/response interaction: The peer sens a request and the backend sends a response (which may be an error message).

The transmission channel of EAP-Kerberos is half-duplex, meaning that the peer and backend take turns sending messages. Each sends one Kerberos frame and is then open for another. In the case of the peer, receiving a Kerberos frame makes him free to send another, and in case of the backend, having receiving a Kerberos frame initiates its handling.

3.2. Format of a Kerberos Frame

A Kerberos frame is defined as a single DER-encoded data structure, so a single tag with length and a value that may hold more DER data. The outermost tag and length are of explicit use to EAP-Kerberos.

The outermost tag MUST be an [APPLICATION n] tag. The tag indicates what meaning the frame has in the context of the Kerberos application [usually according to [RFC4120]]. The tag may facilitate relaying decisions, especially in the backend authentication server. Finally, the tag indicates a syntax that the entire Kerberos frame MUST adhere to.

The outermost length is used in the collection of a Kerberos frame that MAY have gotten fragmented over multiple EAP frames. The first

EAP frame to transport a Kerberos frame holds the outermost tag and length and a failure to contain both within this first message MAY be treated as a fatal error.

As a result of these definitions, Kerberos frames can be concatenated; the DER encoding clearly marks where one Kerberos frame ends and the next one starts. This does not mean that such concatenation is permitted within an EAP frame; in fact, it is considered a fatal error if an EAP frame contains excess bytes after the end of a Kerberos frame.

3.3. Encapsulation of Kerberos Frames in EAP

A Kerberos frame is cut into one or more non-empty fragments that lead to EAP frames that fit in the MTU for EAP frames. The EAP-Kerberos protocol consists of such individually encapsulated fragments, as well as acknowledgement packages.

To enapsulate a Kerberos fragment, the standard EAP packet header [Section 4 of [RFC3748]] is prepended, consisting of the fragments's Code, Identifier and Length. Note that it is possible to include a Kerberos fragment with a Success or Failure EAP frame that holds the Success or Failure code, not just for Request or Response codes.

The Length field holds five more than the non-zero number of bytes of the contained Kerberos fragment, and it should be noted that any further bytes count as padding in the data link layer, so these should not be taken into account when reconstructing a Kerberos frame from fragments in EAP frames. EAP packets with Length set to 1 hold no Kerberos data but are instead used to acknowledge the reception of traffic up to the given Identifer.

The Type field is always set to the value TBD assigned by IANA as the Method Type for EAP-Kerberos.

When a Kerberos frame falls apart into multiple fragments, then all these fragments are submitted individually, with an Identifier byte value that increments from the value of its first packet. This is useful on the receiving end, where the portions must be recollected. When it transpires that EAP frames were not received in good order, they MUST be retransmitted. Indications of such ill-received EAP frames are received through special acknowledgement EAP frames. The Identifier starts from 0 for each new Kerberos frame, so the initial EAP frame for a Kerberos frame is easily recognised.

When encapsulating a Kerberos fragment, the EAP frame always has a Length over 5. There is an additional EAP frame used to acknowledge having received prior frames, and this is a frame with Length equal

to 5. It SHOULD only be sent on the last frame before an apparent gap in the frame sequence, and in addition it MUST be sent on the last frame that completes the reconstruction of a Kerberos frame. Acknowledgement packets replicate the Identifier tag from the last EAP frame that they received in good order. TODO: Is this possible, or MUST it be a roundtrip to satisfy carriers such as RADIUS?

Only when a complete Kerberos frame has been reconstructed on the receiving end, is it possible to send the next message in the opposite direction. The backend authentication server can start to build a response after receiving a Kerberos frame, and the authenticating peer can choose if it wants to send another request in a Kerberos frame.

4. Component Support for EAP-Kerberos

The following subsections indicate what support is needed for EAP-Kerberos to be used.

4.1. Peer Support for EAP-Kerberos

In EAP terminology, the (authenticating) peer is the endpoint that wants to be authenticated by the Authenticator. It must have explicit support for EAP-Kerberos.

When the peer has no valid ticket granting ticket, or when it prefers to use another for EAP-Kerberos, then it should first get one using the AS exchange [Section 3.1 of [RFC4120]].

When the peer has no valid service ticket, or when it prefers to use another one, for EAP-Kerberos, then it should get one (or multiple) using the TGS exchange [Section 3.3 of [RFC4120]].

Finally, the peer will want to authenticate with the backend server. To do that, it uses the service ticket in a double AP exchange [Section 3.2 of [RFC4120]]:

1. The first AP request [Section 5.5.1 of [RFC4120]] sent from peer to backend includes no cksum, but supplies a randomly generated seq-number in its Authenticator. It does not include a subkey.

2. The first AP response [Section 5.5.2 of [RFC4120]] sent from backend to peer contains a subkey and a seq-number field that is one more than the seq-number field in the first AP request. The subkey is stored by the peer as its master key.

3. The second AP request sent from the peer to the backend includes a cksum computed over the EncAPRepPart of the first AP response,

after it has been decrypted into a DER structure. The seq-number field is one higher than the seq-number field in the first AP response. The hash assures the peer's ability to decrypt, and thus ownership of the key; the seq-number may help to assure extra protection from replay. The second AP request additionally holds a subkey that the backend should henceforth be used by the backend. The second AP request is encrypted with the subkey found in the first AP response.

4. The second AP response sent from backend to peer includes no subkey but it does hold a seq-number which is one higher than the seq-number from the second AP request. The second AP response is encrypted with the subkey found in the second AP request.

This exchange establishes mutual authentication between the peer and backend.

Peers SHOULD support at least the above variations of Kerberos frame flow, but they MAY attempt to do more; specifically, it may attempt to perform other AS, TGS and AP exchanges than what is strictly needed for authentication to commence. All this must be done before the final authentication step is made.

4.2. Authenticator Support for EAP-Kerberos

In EAP terminology, the authenticator is the part of the infrastructure that wants to know if the authenticating peer can be granted access. It often coincides with the Network Access Server in the customary EAP use case of granting network access.

The authenticator SHOULD NOT modify EAP-Kerberos messages that it passes between peer and a backend authentication server. Specifically, it MUST NOT filter on [APPLICATION n] tags of Kerberos frames.

In summary, there is nothing that needs to change to an authenticator to be able to pass EAP-Kerberos.

4.3. Backend Support for EAP-Kerberos

In EAP terminology, the backend (authentication server) is the endpoint for EAP that makes the decision to grant or deny access to the authenticating peer. It is usually a server that runs an AAA protocol such as RADIUS or Diameter.

To facilitate EAP-Kerberos, the backend reconstructs Kerberos frames from EAP frames, and either handles them locally or relays them as permitted by configured policy.

It is RECOMMENDED for backends to facilitate AS and TGS requests [Section 5.4.1. of [RFC4120]] from the peer, and relay those to a KDC. In both cases, the backend SHOULD perform KDC Discovery [Section 7.2.3 of [RFC4120]] to find the KDC, based on the realm field in the request. It relays the Kerberos frame to this discovered KDC, and awaits a reply (or produces a timeout error to replace it). The reply is relayed back to the peer as a Kerberos frame encapsulated in EAP frames.

The backend MAY additionally relay AP requests [Section 5.5.1 or [RFC4120]] to servers when configured to do so; in this case the realm and sname fields of the contained ticket are helpful to locate the target for relaying. The applications KRB-SAFE, KRB-PRIV and KRB-CRED have no indication of the protocol being used, so their relaying is less likely. What this means is that the facility for relaying AP requests is rather limited, and only suitable for situations that require nothing but the AP requests themselves; it may be useful to exchange key materials in the backend infrastructure, such as end-to-end key material.

The backend MUST accept AP requests directed to a service name of its own. Such requests are processed under the following rules:

- Upon arrival of an AP request, the encrypted part is decrypted, by default with the encryption contained in the ticket. When a subkey has been previously supplied to the peer in the current EAP-Kerberos session, then this replaces the default encryption key from the ticket.

- An AP request from the peer has a seq-number field, and the response MUST have a seq-number increased by one.

- When the peer has not been sent a subkey before, one is generated from a good random source, and sent in the AP response. This is done only once per EAP-Kerberos session.

- Every AP request receives an AP response or a KRB-ERROR; this is even true if the AP request leads to the deciding one. The AP response is encrypted with the key in the ticket by default; when a subkey has just been supplied by the AP request from the peer, then this is used instead.

- When timing in the AP request diverts too far from the clock time of the backend, then an error is sent instead of an AP response. It is common to permit a time window of about five minutes around the backend clock time.

An AP request leads to the final decision of either Failure or Success when a subkey has been sent to the peer, when the peer just sent a subkey and cksum in the encrypted part of the AP request. The decision is Failure, except when all the following requirements are met and the decision therefore becomes Success:

- The timing information diverts is close enough to the clock time of the backend.
- The seq-number is one more than the seq-number that was sent to the peer in a preceding AP response sent during this EAP-Kerberos session;
- The cksum is a correctly computed checksum over the DER-encoded bytes that hold the EncAPRepPart after decryption, taken from the foregoing AP response in the current EAP-Kerberos session.

The backend MUST send exactly one Kerberos response to any Kerberos request that it receives. If it receives a request encapsulated in EAP that it does not handle, then it MUST reply with a KRB-ERROR message [Section 5.9.1 of [RFC4120]] with a suitable error code [Section 7.5.1 of [RFC4120]], and/or trigger a fatal error.

5. Established Master Key

When an EAP-Kerberos session ends in Success, then both sides have provided a subkey to the other side, to be used for future communication towards them. Upon success, these keys MAY be used to initiate any further keying requirements between the peer and the infrastructure setup by the backend.

We do not prescribe the manner in which this is done, or how the master key is used to construct session keys, but leave it to the general remark that session keys should be derived in a manner that does not permit derivation of the master key or other session keys derived from it.

6. Efficiency Considerations

Among the uses of EAP are applications where no IP address has been obtained by the peer, such as gaining network access over PPP or Ethernet protocols with 802.1x protection. In such networks, it may be required to bootstrap Kerberos tickets that would be needed for EAP-Kerberos. This is why AS and TGS exchanges may be passed over EAP, and why the backend server may support such exchanges by relaying such traffic to a KDC.

Once access to a network has been granted, there may be an additional need to encrypt traffic, and facilitate authentication. For such exchanges, there may be a use for additional Kerberos exchanges. This is why this specification leaves room for such additional exchanges, without enforcing them.

7. Privacy Considerations

The EAP-Kerberos exchange reveals some information that may be considered private. This may be a privacy concern in infrastructures that pass the EAP-Kerberos data through third party network components. It is possible facilitate privacy for any intermediate EAP relays by sending EAP-Kerberos within EAP tunneling mechanisms, such as EAP-TTLS [RFC5281] or EAP-FAST [RFC4851].

8. Security Considerations

Tunneling is not required for reasons of security; Kerberos frames are designed to travel over untrusted networks.

When EAP-Kerberos travels in the company of attributes descriptive of the peer, it is important to understand that nothing that EAP-Kerberos will do can assign any validity to these attributes. The "innert" identity as mentioned in Kerberos packets is what is validated. Any "outer" identity as mentioned in such companion attributes is more inclined towards routing than to identification.

The peer and its KDC establish end-to-end trust on the basis of their pre-shared secret. This is subsequently used to infer trust on derived keys and tickets. For the intermediaries (namely, the authenticator and even the backend authentication server) the secret is not known, and so they may need extra bases for trust in the established relationship. Specifically, where the lookup of a KDC does not require DNSSEC to secure the client, it will often be instrumental for such intermediate pieces of the infrastructure to establish trust in the authenticity of the KDC for a particular realm or domain name. Note however that the eventual use of the AP exchange between the peer and its backend should imply an alternative trust basis between these two players, through either a shared KDC or two that have a secure realm-crossover relationship. This leaves the authenticator as the main party to protect itself; note that this party can be especially sensitive when it takes action based on companion attributes that travel back with an EAP-Kerberos Success.

9. IANA Considerations

This specification requests the registration of a Method Type in the Extensible Authentication Protocol (EAP) Registry, from the range granted through the Designated Expert with Specification Required procedure. IANA has assigned TBD to the EAP method specified herein.

10. References

10.1. Normative References

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <http://www.rfc-editor.org/info/rfc3748>.

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, DOI 10.17487/RFC4120, July 2005, <http://www.rfc-editor.org/info/rfc4120>.

10.2. Informative References

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, DOI 10.17487/RFC2865, June 2000, <http://www.rfc-editor.org/info/rfc2865>.

[RFC4851] Cam-Winget, N., McGrew, D., Salowey, J., and H. Zhou, "The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST)", RFC 4851, DOI 10.17487/RFC4851, May 2007, <http://www.rfc-editor.org/info/rfc4851>.

[RFC5281] Funk, P. and S. Blake-Wilson, "Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)", RFC 5281, DOI 10.17487/RFC5281, August 2008, <http://www.rfc-editor.org/info/rfc5281>.

[RFC6733] Fajardo, V., Ed., Arkko, J., Loughney, J., and G. Zorn, Ed., "Diameter Base Protocol", RFC 6733, DOI 10.17487/RFC6733, October 2012, <http://www.rfc-editor.org/info/rfc6733>.

Author's Address

Rick van Rein
ARPA2.net
Haarlebrink 5
Enschede, Overijssel 7544 WP
The Netherlands

Email: rick@openfortress.nl