Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 15, 2018

R. Van Rein
ARPA2.net

August 14, 2017

Simple Authentication and Security Layers over the Extensible Authentication Protocol (EAP-SASL)
draft-vanrein-eap-sasl-00

Abstract

This specification permits SASL authentication as an application of EAP. This enhances SASL with several new protocols over which it can be used. It enhances EAP with application-level credentials and mechanisms.

Status of This Memo

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 IKEv2 [Section 3.16 of [RFC7296]], PPP [Section 3.2 of [RFC3748]], 802.1x [Section 3.3 of [RFC3748]] or PANA [RFC5191] 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. Note that PPP is used in technologies like PPPoE, PPPoA, L2TP and PPTP; and that 802.1x is used in EAPOL authentication for wired networks, as well as for wireless ethernet where it is called WPA2; IKE is used to negotiate IPsec security associations. In short, EAP is available in most places that grant network access.

Although often used with password-based policies, EAP may also be used with more advanced cryptographic approaches. What this specification adds, is a facility for the Simple Authentication and Security Layer, with a focus on the most generally used facilitation of client authentication. Both EAP and SASL follow a request/ response interaction, which makes their integration into EAP-SASL relatively straightforward.

Typical applications of EAP are network access, while SASL is more oriented towards end-user applications. There is no reason however, why the potential of SASL mechanisms should be held back from EAP. What EAP stands to gain from this is an independently standardised and implemented set of authentication mechanisms; depending on implementation choices, these may be made to share credentials for end-user applications, which can be helpful when networks move into

the user view, such as is the case with VPNs and single-user network logon (perhaps from a single-user machine). What SASL stands to gain is the ability to be carried over widely used AAA backend protocols such as RADIUS and Diameter. When a site is standardising its authentication on SASL, it is possible for both network access and end-user applications to isolate authentication sequences and relay them to a shared AAA backend. This facilitates centralised management of identities and credentials.

Some special attention is needed for one of the SASL mechanisms, namely Kerberos5 over GSSAPI. This mechanism uses short-lived credentials, which may mean that a bootstrapping sequence is needed so these can be setup. The work in progress on IAKERB [I-D.ietf-kitten-iakerb] enhances GSSAPI with just this facility. This is more general, and therefore better, than earlier work done on EAP-Kerberos [I-D.vanrein-eap-kerberos]. An explicit method for Kerberos over EAP is an improvement over current-day implementations that use the PAP method to pass the client password over RADIUS which then addresses a KDC by authenticating on behalf of the user. It should be noted that such submission of user passwords contradicts Kerberos security design assumptions.

There is a SASL method for EAP over GSSAPI [RFC7055], which could be combined with this specification to form stacks like EAP-SASL-GSSAPI-EAP and SASL-GSSAPI-EAP-SASL, both of which seem useless and are unintended. These stacks may however occur as a result of abstraction layers that are unaware of lower or higher abstraction layers. Although not a sign of good design, this specification cannot forecast all possible uses, so it limits itself to stating that such stacks are NOT RECOMMENDED.

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

Mechanism:	SASL mechanisms with operator approval
Ciphersuite negotiation:	Only with supporting SASL mechanisms
Mutual authentication:	Only with supporting SASL mechanisms
Integrity protection:	No
Replay protection:	Only with supporting SASL mechanisms
Confidentiality:	No
Key derivation:	Yes, except with plaintext passwords
Key strength:	Follows SASL credential entropy
Dictionary attack protect:	Follows SASL credential entropy
Fast reconnect:	No
Cryptographic binding:	No
Session independence:	Yes, ensured through fresh random salts
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 the EAP-Message attribute holding an EAP packet; the contents of this User-Name would be the outer identity.

Only the inner identities of EAP-SASL relate to authenticated identities, at least when EAP approves the exchange. This remains true when EAP-SASL is wrapped inside of EAP-TTLS, as described below.

An outer identity may be added for routing purposes alone, where the realm part of the User-Name serves to indicate a backend to route to. In fact, for reasons of privacy, the outer identity often lacks the user name and may look like @example.com.

3. Transporting SASL Frames over EAP

Messages from the SASL client to the server are sent in an EAP packet with Code set to Request. What is called a client in SASL is referred to as a peer in EAP. Challenges from the server to the SASL client are sent in EAP packets with Code set to Response. Both the EAP packets, which shall be referred to as EAP Request and EAP Response respectively, will set Type to TBD, as assigned by IANA. The Type-Data field in both an EAP Request and EAP Responses is referred to as EAP Payload below. The names EAP Success and EAP Failure will be used to refer to an EAP packet with Code set to Success and Failure, respectively.

The SASL specification [RFC4422] is often used with base64-encoding of binary data, to avoid problems of textual protocols. EAP is a binary protocol, so it can carry binary content directly in EAP. For this reason, no base64 or other mapping to text will be used.

The EAP Payload consists of 20 bytes followed by the binary data for the SASL mechanism, the latter of which may be 0 bytes long. The first 20 bytes hold the SASL mechanism name or an instruction. Instructions are easily recognised because they start with an asterisk (U+002a). Note that instructions are not valid SASL mechanism names; they are used to expand EAP with specific semantics of EAP-SASL. The SASL mechanism name or instruction starts in the first byte of the EAP Payload and is padded with space characters (U+0020) to fill up the 20 bytes. Note that according to the syntax

of SASL names [Section 3.1 of [RFC4422]], 20 bytes can hold the longest SASL name.

The instruction *RESET can be sent by EAP peers to terminate a current EAP session, if any; the EAP server responds with EAP Failure, which only counts as a failed session inasfar as one was currently active. This form of termination is never used by the EAP server, which instead sends EAP Failure in its next EAP Response message. The *RESET instruction is never followed by SASL mechanism data bytes. The instruction SHOULD be used when the EAP Lower Layer is a multiplex of EAP links without explicit link ends, and it MAY be used when it uses a connection-less transport without any certainty about the remote peer's state (such as after a software restart on either end). When no EAP interaction is taking place, the EAP Payload with this instruction has no effect. On connection-less EAP transports, this instruction may be used to safely start an interaction after one side is restarted while the other may still keep state. When an EAP Payload with this instruction is used, both its sender and recipient MUST discard any EAP-related state and forward the error to any other protocol layers that may depend on it.

The instruction *DEFRAGMENT is used when an EAP Payload cannot be sent in one whole; any but the last EAP Payload is sent with this instruction. It may be sent by the EAP peer or server when the MTU available for EAP is insufficient to carry a full EAP Reqeust or EAP Response. The size of the data following this instruction and its padding to 20 bytes MUST be non-zero and should make the EAP Payload almost as large as the MTU will permit. Upon receiving an EAP Payload with this instruction, it is held so that the next EAP Payload may be attached to it; the reconstituted EAP Payload will have its SASL mechanism name or instruction set to the first EAP Payload to follow without the *DEFRAGMENT instruction. This reconsituted EAP Payload is then used instead of its constituent components, and processed as had it been sent without transport fragmentation. The receiving party requests continuation with an EAP Request or Response (as implied by their role) with the *CONTINUE instruction and no appended bytes.

The instruction *RANDOMSALT may be exchanged once before EAP Success or EAP Failure. It is initially sent by the EAP peer, and results in the same instruction in an EAP server response. Each party is allowed to send as much random bytes as it likes, but 16 bytes is the REQUIRED minimum and no more than the size of MSK/EMSK that could be generated from it, which means a maximum of 128 bytes under this specification.

The first EAP Payload that names the SASL mechanism may just be 20 bytes in size, in which case its optional initial data is considered

absent. When a *DEFRAGMENT instruction precedes it, the optional initial data is considered present, even when it is 0 bytes long. SASL requires distinction between empty and absent initial data [Section 4 of [RFC3748]], which is implemented by these rules.

The instruction *LISTMECHANISMS enables the EAP peer to obtain a list of server-supported SASL mechanism names. The EAP Request with this instruction adds no data bytes; the EAP Response with this instruction adds supported SASL mechanism names separated by a space character (U+0020). The use of this instruction is optional, at the discretion of the peer. When sent, this exchange precedes the customary information.

Any *DEFRAGMENT instructions preceding an EAP Success count as the optional data that a SASL mechanism may receive alongside a successful finish. This even applies when the preceding instructions provide 0 bytes. The absence of preceding *DEFRAGMENT instructions cause an EAP Success to be delivered to the EAP peer without such additional data. Note the distinction between absense of additional data and empty but present This slightly contrived mechanism can fit carrier protocols that allow only one EAP Payload at a time, notably PPP [Section 2.2 of [RFC2284]] as well as others that depend on the EAP Success to be reported over EAP and not merely be derivable from this EAP-SASP instruction. When EAP Failure is reported by the server, any preceding *DEFRAGMENT instructions from the server MUST be ignored, and silently discarded before delivery to the EAP peer.

4. Interactions with the EAP Lower Layer

SASL requires a number of specifications from the protocol that embeds it. Some of these can be resolved in EAP in a generic manner, as was done in the preceding section; this section pushes a few requirements to the layers below EAP, which we shall refer to as the EAP Lower Layer, for which some possibilities are enumerated in the Extensible Authentication Protocol (EAP) Registry maintained by IANA, while others such as RADIUS and Diameter are only defined by incorporation of EAP in their messages.

The Identity field in an EAP header [Section 4 of [RFC3748]] is the only mechanism that can distinguish EAP packets, and it is used to match a Response to a Request. Though it might be used to allow 128 or perhaps even 256 simultaneous EAP interactions, this is neither forbidden nor specified herein. Instead, the RECOMMENDED place to implement concurrency is in the EAP Lower Layer; a PPP stream is always encapsulated into a session, as is the case for L2TP and PPPoE and even dialup networking; a RADIUS stream can include an EAP-Session-Id; a Diameter stream uses a Session-ID to connect parts of the same flow; IKEv2 uses SPIs; PANA message headers have a Session

Identifier field. Adding an interpretation to the Identity field for reasons of concurrency would add complexity, but not functionality. In line with this reasoning, multiple SASL authentications within the same EAP session are not supported.

SASL requires a service name to be specified for use with GSSAPI. This service name is easy to establish when protocols serve a specific application, but EAP is more general. Therefore, the responsibility of selecting a service name must in general be specific to the EAP Lower Layer. The name TBD:net is allocated as a default service name for the EAP Lower Layer protocols 1-7 and 9, as registered by IANA. The EAP peer drives this choice. When carried over RADIUS or Diameter, the EAP-Lower-Layer attribute [Section 7.2 of [RFC6677]] SHOULD be used. For EAP Lower Layer protocol 8 (GSS-API), this specification cannot assign a default service name, because it is another generic lower layer. Also, it may be better to avoid the SASL method GSSAPI when GSSAPI is the EAP Lower Layer.

SASL requires a standard syntax and semantics, as well as normalisation rules, for authorisation identifiers. In general, this depends on the EAP Lower Layer. We can provide a default mechanism, however, which is of use to customary EAP Lower Layers such as PPP and 802.1x. This is either the format of a Fully Qualified Domain Name [Section 5.2 of [RFC1594]] or a Network Access Identitifier [RFC7542].

TODO: channel binding, like in GS2-KRB5-PLUS and SCRAM-SHA1-PLUS requires a notion of the TLS channel; relaying it to a backend over EAP is not helpful; an attribute such as the NAS-Port-Instance may help?

5. Key Derivation

There are two mechanisms that may be used for key derivation; either directly using the SASL credentials, or by using an EAP-TTLS wrapper around EAP-SASL. When both are used, EAP-TTLS is the RECOMMENDED choice on account of its stronger security.

5.1. Keying under EAP-SASL

Keying under SASL uses what shared secret is available in order to generate MSK/EMSK [RFC5247] for use in the EAP Lower Layer. To enable this additional function, the shared secret MUST NOT be passed over EAP-SASL in an unprotected form, not even when protected with EAP-TTLS, meaning that the SASL PLAIN and OAUTHBEARER mechanisms are barred. Some form of active use of the credential MUST pass over EAP-SASL, meaning that the SASL ANONYMOUS and EXTERNAL mechanisms are also barred. Furthermore, the EAP-SASL method MUST have exchanged at

least 16 bytes from each side in precisely one *RANDOMSALT instruction exchange.

SASL mechanisms that can negotiate a security layer are RECOMMENDED to use this facility to find a sesssion key for key generation under SASL; other SASL mechanisms may have to use a shared key that is fixed as the session key. One SASL mechanism that SHOULD negotiate a session key when used is GSSAPI with Kerberos5.

The session key is used in a variation on the EAP-TTLS computation:

```
KeyMaterial = PRF-128 (eap_sasl.session_key,
                       "EAP-SASL keying material",
                       peer.RANDOMSALT + server.RANDOMSALT)

MSK  = KeyMaterial [ 0.. 63]

EMSK = KeyMaterial [64..127]
```

The PRF-128 function is as used with EAP-TTLS [Section 7.8 of [RFC5281]]. Note the reversed order of the random material relative to TLS; as for EAP-TLS and EAP-TTLS, this intends to avoid clashes with similar TLS computations. The EMSK may be used to derive root keys [RFC5295].

5.2. Keying under EAP-TTLS

Some SASL mechanisms cannot reliably be sent in plaintext; in such cases, it is customary to run the containing protocol in a secure transport such as TLS. There is an EAP mechanism to do just that, namely EAP-TTLS [RFC5281]. The EAP-SASL flow can be embedded in EAP-TTLS if such is required.

Wrapping EAP-SASL inside EAP-TTLS is not only of interest for protecting the credential, but also in assuring that the response is sent to an authentic server, thus mitigating man-in-the-middle attacks. Finally, EAP-TTLS can be useful because it standardises a key that can be used to encrypt further traffic, for instance under the WPA2 technology.

Finally, when EAP-TTLS wrapping is used, there is the option of using Keying Material Export [RFC5705] to derive a key. Unlike the MSK/EMSK that EAP-TTLS shares with the EAP authenticator to establish one-hop encryption, this constitutes a key that reaches out to the site that performed authentication. Such applications use the ASCII label "EAP-SASL key derivation" (without the quotes). The context value is only provided when EAP-SASL has exchanged *RANDOMSALT, in

which case it is set to the concatenation of the peer and server random salt bytes.

6. Efficiency Considerations

EAP is a lock-stepped request/response protocol, which means that there are no window buffers, and so efficiency can be low, especially when EAP-SASL is run over EAP-TTLS. This is agravated in setups with a backend AAA server, especially when the backend is dynamically switched to a remote site. Since EAP is mostly used to bootstrap network connections, rather than consistently recurring events, this is usually considered acceptable.

Some EAP-SASL mechanisms or encapsulations may derive an end-to-end secret key that cannot be observed by intermediates. This may speed up further processing, for instance for the setup of IPsec shared keys.

SASL is present in many protocols, each of which could be candidates for backend authentication. Many protocols (like IMAP and SMTP) do not allow reuse of connections for multiple authentications. LDAP does allow such reuse, and overcomes MTU-caused fragmentation, but only one LDAP bind interaction can be active at any time. The best performance is to be expected from the multiplexed authentication sessions over AAA protocols. Of these, Diameter over SCTP is likely the most efficient, because it (1) avoids head-of-line blocking by sending out-of-order, (2) avoids resend logic and timers with reliable delivery, (3) avoids fragmentation by allowing large user messages, (4) handles resends at the protocol level, and can notice intermediate frames being dropped, (5) can combine multiple small messages in one network packet.

7. Privacy Considerations

The EAP-SASL mechanism is as private as SASL is. So, when a user identity is revealed in plaintext SASL, then it will also be visible in plaintext EAP-SASL. A layer of EAP-TTLS can remedy any problems with this.

In addition, EAP-SASL may be transported with a so-called outer identity. If this is the case, then additional data may leak from there too. The customary approach is then to avoid mentioning the username portion, but just the realm, as in @example.com User-Id attributes.

8. Security Considerations

Anything that cannot travel securely in plaintext over SASL, also cannot travel securely over EAP-SASL. Where needed, a layer of EAP-TTLS can be used to remedy that.

SASL mechanisms are not only subjected to public showing of credentials, but also to the danger of entering in a challenge/response interaction with an unverified peer, which may then function as a man in the middle. Such men are usually called Eve. A wrapper of EAP-TTLS can remedy this, by supplying end-to-end server authenticity.

Whether plaintext suffices for EAP-SASL depends largely on the intermediate network; when routing to an external network it is almost certainly not a good idea but when routing to a nearby AAA backend within a secure network premises or over a secure AAA backlink it can be made secure.

When producing MSK/EMSK from EAP-SASL, it is vital to have good entropy from all the available places: the session key, the EAP peer and EAP server should all provide an ample amount of entropy. The *RANDOMSALT provided by EAP peer and server helps each side to direct scattering of the MSK/EMSK, and thereby influence that other parties could attempt replay attacks; but regardless of the quality and size of these *RANDOMSALT fragments, the session key is still subject to password attacks upon observation of the MSK/EMSK and must therefore be carefully selected; this is especially a concern when the session key is just a shared secret, such as a password, which may be used without change over a prolonged period.

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.

This specification defines an additional entry in the registry "Generic Security Service Application Program Interface (GSSAPI)/Kerberos/Simple Authentication and Security Layer (SASL) Service Names" namely:

Service Name:	net
Usage:	authentication for network access
Reference:	TBD:this specification

This specification defines an entry in the TLS Exporter Label registry, referencing this specification, namely: EAP-SASL key derivation

10. References

10.1. Normative References

[I-D.ietf-kitten-iakerb]
Kaduk, B., Schaad, J., Zhu, L., and J. Altman, "Initial and Pass Through Authentication Using Kerberos V5 and the GSS- API (IAKERB)", draft-ietf-kitten-iakerb-03 (work in progress), March 2017.

[RFC1594] Marine, A., Reynolds, J., and G. Malkin, "FYI on Questions and Answers - Answers to Commonly asked "New Internet User" Questions", RFC 1594, DOI 10.17487/RFC1594, March 1994, <http://www.rfc-editor.org/info/rfc1594>.

[RFC2284] Blunk, L. and J. Vollbrecht, "PPP Extensible Authentication Protocol (EAP)", RFC 2284, DOI 10.17487/RFC2284, March 1998, <http://www.rfc-editor.org/info/rfc2284>.

[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>.

[RFC4422] Melnikov, A., Ed. and K. Zeilenga, Ed., "Simple Authentication and Security Layer (SASL)", RFC 4422, DOI 10.17487/RFC4422, June 2006, <http://www.rfc-editor.org/info/rfc4422>.

[RFC5247] Aboba, B., Simon, D., and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", RFC 5247, DOI 10.17487/RFC5247, August 2008, <http://www.rfc-editor.org/info/rfc5247>.

[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>.

[RFC5295] Salowey, J., Dondeti, L., Narayanan, V., and M. Nakhjiri, "Specification for the Derivation of Root Keys from an Extended Master Session Key (EMSK)", RFC 5295, DOI 10.17487/RFC5295, August 2008, <http://www.rfc-editor.org/info/rfc5295>.

[RFC5705] Rescorla, E., "Keying Material Exporters for Transport Layer Security (TLS)", RFC 5705, DOI 10.17487/RFC5705, March 2010, <http://www.rfc-editor.org/info/rfc5705>.

[RFC6677] Hartman, S., Ed., Clancy, T., and K. Hoeper, "Channel-Binding Support for Extensible Authentication Protocol (EAP) Methods", RFC 6677, DOI 10.17487/RFC6677, July 2012, <http://www.rfc-editor.org/info/rfc6677>.

[RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542, DOI 10.17487/RFC7542, May 2015, <http://www.rfc-editor.org/info/rfc7542>.

10.2. Informative References

[I-D.vanrein-eap-kerberos]
Rein, R., "Kerberos in the Extensible Authentication Protocol (EAP-Kerberos)", draft-vanrein-eap-kerberos-00 (work in progress), March 2016.

[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>.

[RFC5191] Forsberg, D., Ohba, Y., Ed., Patil, B., Tschofenig, H., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA)", RFC 5191, DOI 10.17487/RFC5191, May 2008, <http://www.rfc-editor.org/info/rfc5191>.

[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>.

[RFC7055] Hartman, S., Ed. and J. Howlett, "A GSS-API Mechanism for the Extensible Authentication Protocol", RFC 7055, DOI 10.17487/RFC7055, December 2013, <http://www.rfc-editor.org/info/rfc7055>.

[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>.

Author's Address

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

Email: rick@openfortress.nl