3GPP TSG SA WG3 Security — S3#21 **S3-010652**

27 - 30 November, 2001

Sophia Antipolis, France

Source: Alcatel

Title: Comments on draft-torvinen-http-eap-01.txt

Document for: Discussion

Agenda item:

INTERNET-DRAFT

Document: draft-torvinen-http-eap-01.txt

Expires: May 2002

J. Arkko

V. Torvinen

Ericsson

A. Niemi

Nokia

November 2001

HTTP Authentication with EAP

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Abstract

This document describes a HTTP authentication scheme using PPP Extensible Authentication Protocol (EAP).

HTTP EAP authentication enables HTTP connections to be authenticated using any of the authentication schemes supported through EAP. EAP performs the authentication without sending the password in the clear text format (which is the biggest weakness of the Basic HTTP authentication scheme, for example).

It is useful for HTTP protocol because it opens up several new authentication schemes without additional specification work. The same benefits can be reached by any other protocols, which apply HTTP authentication, such as Session Initiation Protocol (SIP).

Table of Contents

Conventions used in this document

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

this document are to be interpreted as described in RFC-2119 [1]

1 Introduction

The HTTP Authentication framework includes two authentication schemes: Basic and Digest [2]. In the Basic scheme, the client authenticates itself with a user-ID and a password for each realm. The Basic scheme is perceived as insecure since the user credentials are transmitted across the public network in a cleartext format. The Digest scheme is based on cryptographic hashes and is consequently perceived as a more secure authentication scheme than Basic, but is limited the use of passwords. See [2] for detailed information about the general HTTP authentication protocol.

The PPP Extensible Authentication Protocol (EAP) is a general protocol for PPP authentication [3]. Even though EAP was originally developed as a link layer protocol, it can also be applied at the application layer. EAP supports multiple authentication mechanism (e.g. smart cards, Kerberos, Public Key, One Time Passwords, and others) and it can, by definition, be easily extended to support new authentication mechanisms [see e.g. 4, 5, 6, 7]. EAP packets are defined in a binary format, and their contents depend highly on the used authentication scheme.

HTTP EAP Authentication Scheme supplements HTTP Authentication with EAP functionality. This opens up several new authentication schemes for HTTP Authentication without additional specification work.

2 HTTP EAP Authentication Scheme

The HTTP EAP Authentication Scheme delivers base64 encoded EAP packets within HTTP Authentication headers (e.g. WWW-Authenticate Response headers and Authorization Request headers). EAP packets include all relevant information about the required authentication scheme, e.g. authentication scheme, packet type (request, response, success or failure) and/or challenge. The content of these packets is up to the chosen EAP authentication scheme.

The progression of an authentication procedure depends also on the chosen authentication mechanism. Typically, the authenticator sends an initial Identity Request followed by one or more Requests for authentication information. The peer sends a Response packet in reply to each Request. As with the Request packet, the Response packet contains a type field, which corresponds to the type field of the Request. The authenticator ends the authentication phase with a Success or Failure packet. See Figure 1.

User agent Server

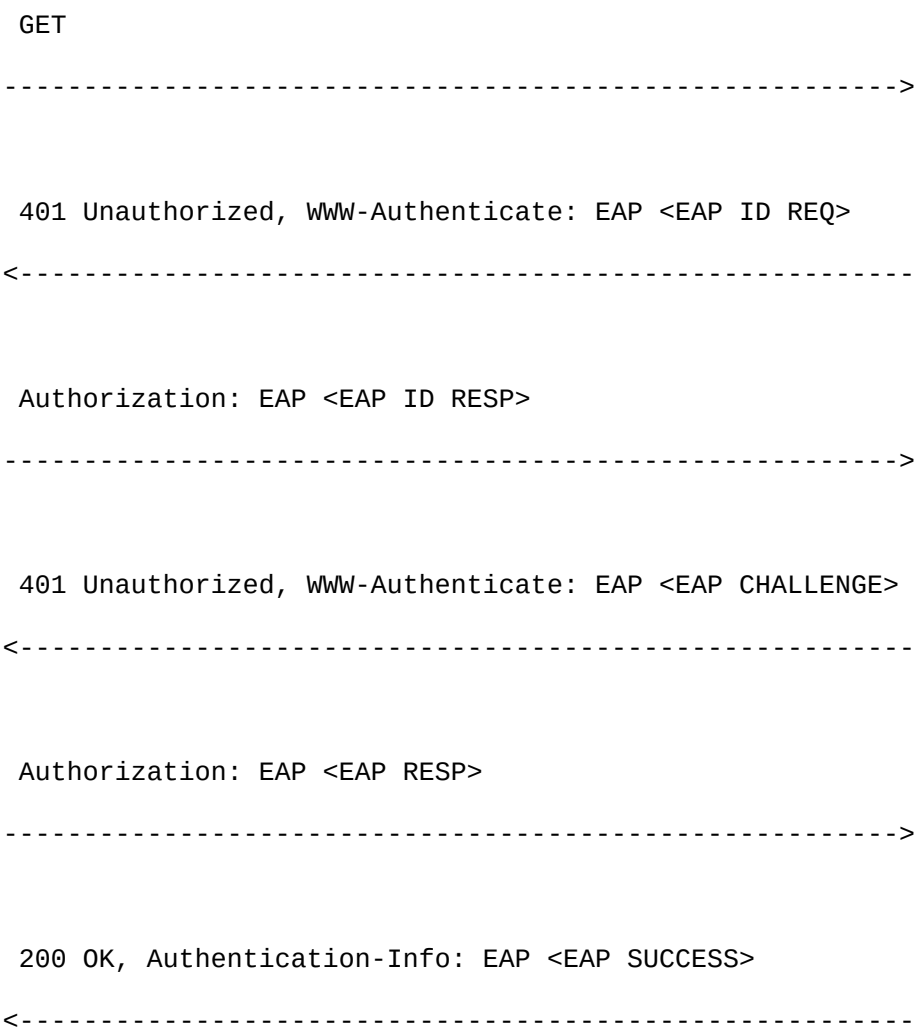


Figure 1. HTTP EAP Authentication message flow

This message flow above represents only the typical situation. Variations of the flow are also possible in the following situations:

- The chosen authentication mechanism requires more than the single challenge-response message pair shown. Any number of message exchanges are allowed here.
- Error situations result in terminating the flow from the server’s side with an error response. This response could be one of 401

Unauthorized, 403 Forbidden, or 407 Proxy Authentication Required. For 401 and 407, the client distinguishes the error situation from the continuation of the EAP exchange by the existence of EAP FAILURE payload, or the lack of any EAP payload.

- Error situations from the client's side result in terminating the communications with the server.
- Certain EAP authentication mechanisms such as [7] allow an optimized flow where identity request does not need to be sent. In these cases, if the client knows it will be demanded EAP authentication, it can include an unsolicited EAP ID RESP already

in the GET message. This would enable the server to start the actual authentication exchange immediately.

- EAP authentication was shown to be run towards the server which responds with 401 Unauthorized responses. It is also possible to run towards a proxy, which responds with 407 Proxy Authentication Required responses.

In this document, we define three new header types for the HTTP

[Alcatel] the above is misleading as these are not new header types (already defined in RFC 2617) but rather new schemes in those existing header types.

authentication framework. These headers, WWW-Authenticate Response Header, Authorization Request Header and Authentication-Info Response Header, are needed for making EAP as an independent HTTP authentication scheme.

2.1 The WWW-Authenticate Response Header

The general HTTP authentication framework uses an extensible, case-insensitive token to identify the authentication scheme. Authentication scheme identifier is followed by a comma-separated list of attribute-value pairs, which carry the parameters necessary for achieving authentication via that scheme.

```
auth-scheme     = token
auth-param      = token "=" ( token | quoted-string )
```

If a server receives a request for an access-protected object without an acceptable Authorization header, the server responds with a "401 Unauthorized" status code, a WWW-Authenticate header and at least one challenge applicable to the requested resource. A Proxy acts in the same way but it uses a "407 Proxy Authentication Required" status code instead.

```
challenge       = auth-scheme 1*SP 1#auth-param
```

The authentication parameter realm is defined for all authentication schemes:

```
realm           = "realm" "=" realm-value
realm-value     = quoted-string
```

The realm value and the canonical root URL of the server being accessed define the protection space.

The realm directive (case-insensitive) is required for all authentication schemes that issue a challenge. The realm value (case-sensitive) is a string, which may have additional semantics specific to the authentication scheme.

For HTTP EAP Authentication, the framework above is utilized as follows:

```
challenge       = "Eap" eap-challenge

eap-challenge   = 1#(realm | eap-param)
```



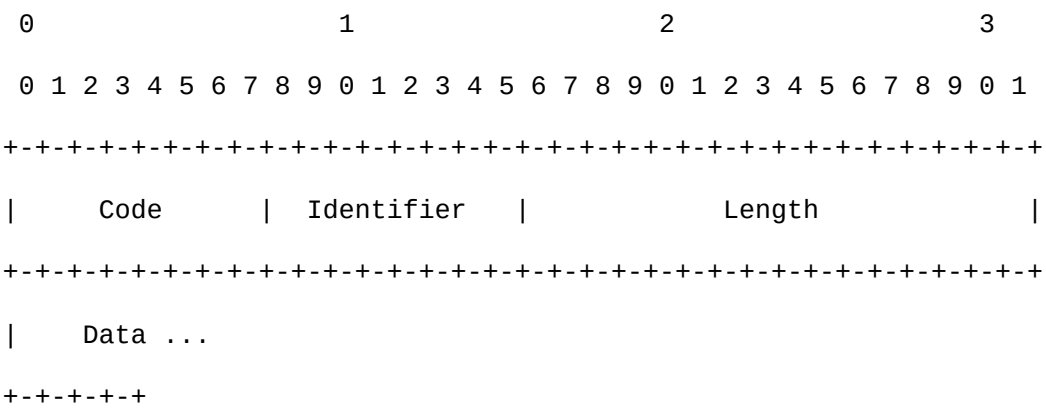
```
realm           = "realm" "=" <"> realm-value <">
realm-value     = token [ "@" token ]
eap-param       = "eap-p" "=" <"> eap-packet <">
eap-packet      = <base64 encoded eap-packet, except
                   not limited to 76 char/line>
```

The realm value SHOULD be globally unique. Proxy servers are RECOMMENDED to use globally unique realm values in order to be able to recognize their set of user credentials in a multi-proxy authentication scenario. Implementations MAY use the form "local-realm@host".

The realm value should be considered as an opaque string, which can

only be compared for equality with other realms on that server. The server will service the request only if it can validate the user credentials for the protection space of the Request-URI.

EAP packets have a general structure consisting of four basic fields: code, identifier, length and data. The Code field is one octet and it identifies the type of the EAP packet. Packet type is either a request, response, success, or failure. The Identifier field is also one octet and it is used for matching responses with corresponding requests. The Length field is two octets and it indicates in octects the length of the whole EAP packet including code, identifier, length and data fields. The Data field is zero or more octets and its format depends on the content of Code field. The example below demonstrates the general structure of EAP packets.



All these fields (Code, Identifier, Length, and Data) are included in the eap-packet in base64 form. Note that since the packets are self-identifying and self-delimiting it is allowed to include multiple EAP packets within one eap-packet, should some EAP mechanism be able to benefit from this.

Example below demonstrates how a WWW-Authenticate Response Header

using EAP authentication would look like:

```
WWW-Authenticate: eap realm="BollyWorld@example.com",
eap-p="QWxh4ZGRpb2jpvcGVuNlctZQ=="
```

where "BollyWorld" is the string assigned by the server to identify the protection space of the Request-URI at server "example.com".

A proxy may respond with the same challenge using the Proxy-Authenticate header field. Then it is especially important to

maintain global uniqueness for the realm values, since a request may have credentials for multiple Proxy-Authenticate challenges.

2.2 The Authorization Request Header

In the general HTTP authentication framework, a user agent that wishes to authenticate itself with an origin server or a proxy MAY do so by including an Authorization header or a Proxy-Authorization header field to the request. The authorization field value(s) consists of credentials containing the authentication information of the client for the realm of the resource being requested. The user agent MUST apply the strongest authentication scheme it understands

and request credentials from the user based upon the corresponding challenge.

```
credentials     = auth-scheme #auth-param
```

For HTTP EAP Authentication, the framework above is utilized as follows:

```
credentials     = "Eap" eap-response
eap-response    = 1#( realm | eap-param )
eap-param       = "eap-p" "=" eap-packet
eap-packet      = <base64 encoded eap-packet, except
                   not limited to 76 char/line>
```

The value of the realm field must be that supplied in the WWW-Authenticate or Proxy-Authenticate response header for the resource being requested.

Example below demonstrates how the Authorization Request Header using EAP authentication would look like:

```
Authorization: Eap realm="BollyWorld@example.com",
eap-p="QWxhZGRpbjpvcGVuIHNlc2FtZQ=="
```

Rules for handling potential user identifiers, passwords, challenges and so on, are defined in EAP protocol [3].

2.3 Authentication-Info Response Header

The Authentication-Info header is used by the server to communicate information back to the client. This can be either the successful authentication in the response, or the continuation of the EAP mechanism.

```
auth-info       = #auth-param
```

For HTTP EAP authentication the framework above is utilized as follows:

```
Auth-info       = eap-packet
```



```
eap-packet      = <base64 encoded eap-packet, except
                   not limited to 76 char/line>
```

Example below demonstrates how the Authentication-Info Response Header using EAP authentication would look like:

```
Authentication-Info: QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

The semantics of Proxy-Authentication-Info follow those of Authentication-Info. Proxy-Authentication-Info is used by proxy servers in conjunction with the "407 Proxy Authentication Required" response, and the consequent client authorization request.

3 Security Considerations

Very little about the security of HTTP EAP Authentication can be stated without knowing the chosen EAP authentication scheme. Generally speaking, depending on the chosen EAP authentication scheme, HTTP EAP is subject to the same security threats as HTTP Authentication. However, there are some general aspects, which SHOULD be considered when analyzing the security of HTTP EAP Authentication:

1) Authentication of clients: All EAP mechanisms authenticate the client, using a method dependent on the mechanism.
2) Authentication of servers: Some EAP mechanisms also perform mutual authentication.
3) Using the strongest authentication mechanism available: Servers and clients accepting multiple authentication mechanisms should be aware of the possibility of ’bidding-down’ attacks where a man-in-the-middle modifies the authentication offers until the peers agree on an easily breakable mechanism. In general, we expect HTTP EAP _based servers to require a predefined authentication mechanism from a particular client in any case, which avoids this problem. For instance, the user data base at a server indicates that user A has a particular public key. The server should then insist on using the EAP TLS [4] mechanism to authenticate the user.
4) Confidentiality: Each EAP mechanism offers its specific protection schemes for the exchanged credentials. For instance, the EAP AKA [7] mechanism sends secure cryptographic hashes rather than cleartext passwords like HTTP Basic Authentication

does, even if both are based on the concept of a shared secret. As in EAP in general, HTTP EAP does not protect against revealing the identity of the client since the EAP ID RESP packets are not encrypted. Confidentiality and integrity of the HTTP requests themselves beyond the authentication parameters is not within the scope of HTTP EAP, but is discussed below under item 7.

5) Replay protection: Each EAP mechanism offers its specific protection schemes for preventing the replay of the credentials. For instance, the EAP AKA mechanism uses a cryptographically strong sequence number scheme. This is in

contrast to the replay possibilities that exist for the HTTP Basic Authentication, and is similar to the use of nonces in the HTTP Digest Authentication.

6) Integrity protection: Again, each EAP mechanism offers its specific protection schemes against a man-in-the-middle modifying the authentication credentials. Mechanisms based on secure hashes prevent any modifications to the authentication parameters themselves. Again, integrity of the HTTP requests themselves beyond the authentication parameters is a separate issue and is discussed below.

7) Integrity and confidentiality protection of the HTTP request itself is also an important issue. Without such protection, it is possible for a man-in-the-middle to read and modify the

actual contents of the request, regardless of any

authentication that was performed

[Alcatel] As explained below, the message is always first sent in clear until authentication has taken place. Therefore, confidentiality of the HTTP request itself is made rather impossible. A possible solution would be for the client to first submit a request containing minimal information, and only resubmit the (complete) request once authentication has taken place and both integrity and confidentiality keys have been derived from the auth scheme (if applicable in the auth scheme).

Currently, there are no such authentication schemes in HTTP authentication, which would fully protect the integrity of HTTP messages. The HTTP Basic Authentication scheme provides no integrity protection. HTTP Digest Authentication provides only limited (and optional) protection. Most header fields and their values could be modified as part of a man-in-the-middle attack. It should also be noted that HTTP EAP does not inherently provide the integrity protection qualities present in Digest, namely the protection of Request-URI and request-method (and possibly the payload).

Even though HTTP EAP Authentication scheme does not include a protection mechanism, it can be used for setting up one. Chosen EAP authentication scheme may be used to generate session keys, which together with some additional security protocol can provide e.g. integrity protection.

However, such protection should include the protection of original HTTP requests as well. This is not trivial because session protection keys are generated during the authentication, which takes place after submitting the request. In practice, full protection is only possible if the request is repeated at the end of the authentication procedure. This is, however, already the behavior in many typical usage situations. For instance, when authenticating a SIP REGISTER message, the authentication procedure takes a few

message rounds, and on each round the REGISTER message is repeated until the session keys are available and the procedure is completed. The last such message can then use integrity protection. Servers that want to avoid man-in-the-middle attacks MUST NOT act on requests until both the authentication procedure has completed and the messages have been received under integrity protection.

4 References

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

2 Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A. and Stewart, L. "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

3 Blunk, L. and Vollbrecht, J. "PPP Extensible Authentication Protocol (EAP)" RFC 2284, March 1998.

4 Aboba, B. and Simon, D. "PPP EAP TLS Authentication Protocol" RFC 2716, October 1999.

5 Aboba, B. "EAP GSS Authentication Protocol" Internet Draft, draft-aboba-pppext-eapgss-08.txt, October 2001.

6 Carlson, J. "PPP EAP SRP-SHA1 Authentication Protocol" Internet Draft, draft-ietf-pppext-eap-srp-03.txt, July 2001.

7 Arkko, J. and Haverinen, H. "EAP AKA Authentication" Internet Draft, draft-arkko-pppext-eap-aka-00.txt, May 2001.

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

2 Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A. and Stewart, L. “HTTP Authentication: Basic and Digest Access Authentication”, RFC 2617, June 1999.

3 Blunk, L. and Vollbrecht, J. “PPP Extensible Authentication Protocol (EAP)” RFC 2284, March 1998.

4 Aboba, B. and Simon, D. “PPP EAP TLS Authentication Protocol” RFC 2716, October 1999.

5 Aboba, B. “EAP GSS Authentication Protocol” Internet Draft,

draft-aboba-pppext-eapgss-03.txt, February 2001.

6 Carlson, J. “PPP EAP SRP-SHA1 Authentication Protocol” Internet Draft, draft-ietf-pppext-eap-srp-01.txt, May 2001.

7 Arkko, J. and Haverinen, H. “EAP AKA Authentication” Internet Draft, draft-arkko-pppext-eap-aka-00.txt, May 2001.

5 Acknowledgements

The authors wish to thank Henry Haverinen and Bernard Aboba for interesting discussions in this problem space.

6 Author's Addresses

Jari Arkko
Ericsson
02420 Jorvas Phone: +358 40 5079256
Finland Email: jari.arkko@ericsson.com

Vesa Torvinen
Ericsson
02420 Jorvas Phone: +358 40 7230822

Finland Email: vesa.torvinen@ericsson.com

Aki Niemi

Nokia Networks

P.O. Box 301

00045 Nokia Group Phone: +358 50 3891644

Finland E-mail: aki.niemi@nokia.com