              Adding Support for Salted Password Databases to EAP-pwd
                      draft-harkins-salted-eap-pwd-03

Abstract

   EAP-pwd is an EAP method that uses a shared password for
   authentication using a technique that is resistant to dictionary
   attack. It included support for raw keys and RFC2751-style double
   hashing of a password but did not include support for salted
   passwords. There are many existing databases of salted passwords and
   it is desirable to allow their use with EAP-pwd.

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 August 22, 2016.

Table of Contents

1. **Introduction**

1.1. **Background**

   Databases of stored passwords present an attractive target for
   attack-- get access to the database, learn the passwords. To
   confound such attacks a random "salt" is hashed with the password and
   the resulting digest is stored, along with the salt, instead of the
   raw password. This has the effect of randomizing the password so if
   two distinct users have chosen the same password the stored, and
   salted, password will be different. It also requires an adversary
   who has compromized the security of the stored database to launch a
   dictionary attack per entry to recover passwords.

   The popularity of password salting means there are a large number of
   such databases deployed and EAP-pwd needs to be able to support them.
   EAP-pwd imposes an additional security requirement on a database of
   salted passwords that otherwise would not exist, see Section 5.

1.2. **Keyword Definition**

   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 [RFC2119].

[2](#). **Salted Passwords in EAP-pwd**

[2.1](#). **Password Pre-Processing**

   EAP-pwd is based on the "dragonfly" password-authenticated key
   exchange (PAKE)-- see [[RFC7664](#)]. This is a balanced PAKE and
   requires that each party to the protocol obtain an identical
   representation of a processed password (see [Section 5](#)). Salting of a
   password is therefore treated as additional password pre-processing
   techniques of EAP-pwd. The salt and digest to use is conveyed to the
   peer by the server and the password is processed prior to fixing the
   password element (see [Section 2.8.3 of [RFC5931]](#)).

   This memo defines eight (8) new password pre-processing techniques
   for EAP-pwd:

   o TBD1: a random salt with SHA-1 ([[SHS](#)])

   o TBD2: a random salt with SHA-256 ([[SHS](#)])

   o TBD3: a random salt with SHA-512 ([[SHS](#)])

   o TBD4: UNIX crypt() ([[CRY](#)])

   o TBD5: OpaqueString and a random salt with SHA-1 ([[SHS](#)])

   o TBD6: OpaqueString and a random salt with SHA-256 ([[SHS](#)])

   o TBD7: OpaqueString and a random salt with SHA-512 ([[SHS](#)])

   o TBD8: OpaqueString and a UNIX crypt() ([[CRY](#)])

   When passing salt, the size of the salt SHOULD be at least as long as
   the message digest of the hash algorithm used. There is no guarantee
   that deployed salted databases have followed this rule, and in the
   interest of interoperability, an EAP peer SHOULD NOT abort an EAP-pwd
   exchange if the salt conveyed during the exchange is less than the
   message digest of the indicated hash algorithm.

   When performing one of TBD5-TBD8 the password SHALL a UTF-8 string
   and SHALL be pre-processed by applying the Preparation and
   Enforcement steps of the OpaqueString profile in [[RFC7613](#)] to the
   password. The output of OpaqueString, also a UTF-8 string, becomes
   the EAP-pwd password and SHALL be hashed with the indidicated
   algorithm.

2.2. The Salting of a Password

For both parties to derive the same salted password there needs to be
a canonical method of salting a password. When using EAP-pwd, a
password SHALL be salted by hashing the password followed by the salt
using the designated hash function:

    salted-password = Hash(password | salt)

The server stores the salted-password, and the salt, in its database
and the client derives the salted-password on-the-fly.

2.3. Using UNIX crypt

Different algorithms are supported with the UNIX crypt() function.
The particular algorithm used is indicated by prepending an encoding
of "setting" to the passed salt. The specific algorithm used is
opaque to EAP-pwd as the entire salt, including the encoded
"setting", is passed as an opaque string for interpretation by
crypt(). The salted password used for EAP-pwd SHALL be the output of
crypt():

    salted-password = crypt(password, salt)

The server stores the salted-password, and the encoded algorithm plus
salt, in its database and the client derives the salted-password on-
the-fly.

2.4. Protocol Modifications

Like all EAP methods, EAP-pwd is server initiated. The server is
required to indicate its intentions, including the password pre-
processing it wishes to use, before it knows the identity of the
client. Because of this the server is forced to select the format of
the database, including how it has been salted, through the routable
portion of the client identity in the EAP-Identity/response. This
limits the ability of the server to support multiple salt digests in
a single password database. To support multiple salt digests, or to
transition from one to another, it is necessary to maintain multiple
password databases and use the routable portion of the client
identity to select one when initiating EAP-pwd.

The server uses the EAP-pwd-ID/Request to indicate the password pre-
processing technique. The client indicates its acceptance of the
password pre-processing technique and identifies itself in the EAP-
pwd-ID/Response. Upon receipt of the EAP-pwd-ID/Response, the server
knows the identity of the client and can look up the client's salted
password and the salt from the database. The server adds the length

of the salt and the salt itself to the EAP-pwd-Commit/Request message
(see Section 2.5).

The server can fix the password element (Section 2.8.3 of [RFC5931])
as soon as the salted password has been looked up in the database.
The client, though, is required to wait until receipt of the server's
EAP-pwd-Commit/Request before it begins fixing the password element.

2.5. Payload Modifications

When a salted password pre-processing technique is agreed upon during
the EAP-pwd-ID exchange the EAP-pwd-Commit payload is modified to
include the salt and salt length (see Figure 1). The server passes
the salt and salt length in the EAP-pwd-Commit/Request; the client's
EAP-pwd-Commit/Response is unchanged and it MUST NOT echo the salt
length and salt in its EAP-pwd-Commit/Response.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |    salt-len   |                                               |
    +-+-+-+-+-+-+-+-+                                               ~
    ~                        Salt           +-+-+-+-+-+-+-+-+-+-+-+-+
    |                                       |                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                       ~
    |                                                               |
    ~                        Element                                ~
    |                                                               |
    ~                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                               |                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               ~
    |                                                               |
    ~                        Scalar         +-+-+-+-+-+-+-+-+
    |                                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
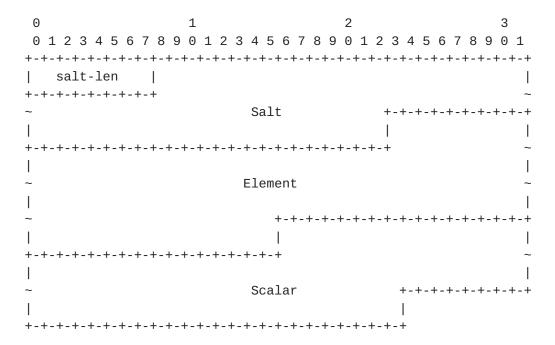
                   Figure 1: Salted EAP-pwd-Commit/Request

The "salt-len" SHALL be non-zero and indicates the length, in octets,
of the salt that follows. When using UNIX crypt for salting, the
salt SHALL be an unterminated ASCII string. For all other salting
algorithms, the salt SHALL be a binary string with no additional
encoding. The Element and Scalar are encoded according to
Section 3.3 of [RFC5931].

Note: when a non-salted password pre-processing method is used, for
example, any of the methods from [RFC5931] the EAP-pwd-Commit payload
MUST NOT be modified to include the salt and salt length.

3. Acknowledgements

Thanks to the Stefan Winter and the eduroam project for its continued interest in using EAP-pwd.

4. IANA Considerations

IANA is instructed to allocate eight (8) values from the "password preprocessing method registry" established by [RFC5931] and replace TBD1, TBD2, TBD3, TBD4, TBD5, TBD6, TBD7, and TBD8 above with the values assigned.

5. Security Considerations

EAP-pwd requires each side to produce an identical representation of the (processed) password before the password element can be fixed. This symmetry undercuts one of the benefits to salting a password database because the salted password from a compromised database can be used directly to impersonate the EAP-pwd client-- there is no dictionary attack needed to recover the plaintext password. While the immediate effect of such a compromise would be the compromised server, the per-user salt would still prevent the adversary from recovering the password, barring a successful dictionary attack, to use for other purposes.

Salted password databases used with EAP-pwd MUST be afforded the same level of protection as databases of plaintext passwords.

6. References

6.1. Normative References

[CRY]   "crypt(3) man page",
           <http://man7.org/linux/man-pages/man3/crypt.3.html>.

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

[RFC5931] Harkins, D. and G. Zorn, "Extensible Authentication
           Protocol (EAP) Authentication Using Only a Password", RFC
           5931, August 2010.

[RFC7613] Saint-Andre, P. and A. Melnikov, "Preparation,
           Enforcement, and Comparison of Internationalized Strings
           Representing Usernames and Passwords", RFC 7613, DOI
           10.17487/RFC7613, August 2015,
           <http://www.rfc-editor.org/info/rfc7613>.

   [SHS]   National Institute of Standards and Technology, , "Federal
              Information Processing Standard Publication 180-4: Secure
              Hash Standard (SHS)", March 2012,
              <http://csrc.nist.gov/publications/fips/fips180-4/
              fips-180-4.pdf>.

6.2. Informative References

   [RFC7664] Harkins, D., Ed., "Dragonfly Key Exchange", RFC 7664, DOI
              10.17487/RFC7664, November 2015,
              <http://www.rfc-editor.org/info/rfc7664>.

Author's Address

   Dan Harkins
   HP Enterprise
   1322 Crossman Avenue
   Sunnyvale, CA 94089-1113
   United States of America

   Email: dharkins@arubanetworks.com