EAP Working Group
Internet-Draft
Category: Standards Track
<draft-funk-eap-md5-tunneled-01.txt>
April 2004

Paul Funk
Funk Software, Inc.

The EAP MD5-Tunneled Authentication Protocol (EAP-MD5-Tunneled)

Abstract

EAP-MD5-Tunneled is an EAP protocol designed for use as an inner authentication protocol within a tunneling EAP protocol such as EAP-TTLS or EAP-PEAP. It is cryptographically equivalent to standard CHAP and the EAP-MD5-Challenge protocol. It can be used inside an EAP tunnel without exposing the system to the type of man-in-the-middle attack which use of CHAP or the original MD5 Challenge protocol is subject to, yet it is capable of being converted to CHAP credentials at the tunneling endpoint for proxy forwarding to legacy AAA servers, with no modification required of the legacy AAA server.

It may also be converted to EAP-MD5-Challenge credentials at the tunneling endpoint for the purpose of proxy; however, the downstream

server that terminates the EAP-MD5-Challenge must be modified to
provide a challenge that meets certain criteria.

Table of Contents

1. Introduction

A number of protocols have recently been proposed that allow legacy
password-based authentication protocols to be securely transported
within a tunnel based on strong one-way authentication of a server
based on its certificate. The intent of these tunneling protocols is
to preserve the utility of the widely-deployed legacy protocols
while securing them against dictionary and other attacks on networks
that are subject to snooping or active interception of connections.

EAP-TTLS and EAP-PEAP are examples of such protocols in the EAP
world; both are based on tunnels created via TLS.

Recent cryptographic analysis has revealed that, while these
protocols do solve a set of security problems, they introduce a new
vulnerability in certain circumstances [MITM]. Because these
protocols do not cryptographically combine session keys derived from
the inner protocol with session keys derived from the outer, it is

possible for an attacker to pose as an authentication server and dupe a client using a legacy protocol without benefit of a tunnel into exchanging protocol payloads with the attacker, which it, now acting as a client, can meanwhile use within a tunneling protocol to authenticate to a legitimate server as if it were the original user.

The following conditions must obtain for such an attack to be feasible: (1) the user must use the same legacy protocol both in tunneled and untunneled modes, (2) the user must use the same credentials (i.e. username and password) in both tunneled and untunneled modes, and (3) the attacker must be able to pose as an authentication server on the network on which the user uses the legacy protocol in untunneled mode.

The IETF EAP working group is studying means to preclude such an attack. It is currently the general opinion of that group that the tunneling protocols can be fixed only when the inner legacy protocols are capable of generating their own session keys [BINDING]. This implies that EAP-TTLS and EAP-PEAP can be made safe against such an attack for inner protocols such as MSCHAP and EAP-SIM, but not for protocols such as CHAP, EAP-MD5-Challenge, or EAP-GenericTokenCard.

In fact, the CHAP and EAP-MD5-Challenge protocols can be made secure against this attack. The EAP-MD5-Tunneled protocol allows a password-challenge authentication to be performed inside the tunnel that can be converted to a CHAP or EAP-MD5-Challenge authentication outside the tunnel for forwarding via RADIUS or other AAA protocol to a legacy backend authenticator.

Yet the EAP-MD5-Tunneled protocol itself is a different protocol than CHAP or EAP-MD5-Challenge, and, while an EAP-MD5-Tunneled authentication can be converted to CHAP or EAP-MD5-Challenge, CHAP or EAP-MD5-Challenge cannot be converted to EAP-MD5-Tunneled. Thus, the necessary conditions for mounting the attack are eliminated.

1.1 Using EAP-MD5-Tunneled to Secure CHAP

CHAP is the most widely used means of authentication in use today for public access to the internet, with a large inventory of user databases and RADIUS servers in support of such authentication. As network providers deploy new networks over media such as radio, they will want to allow their users to continue using the same CHAP protocol with the same credentials against the same RADIUS servers, yet they will want to also take advantage of tunneling protocols to secure user credentials over easily-eavesdropped networks. Use of EAP-MD5-Tunneled allows such new deployments without introducing a new security risk.

When EAP-MD5-Tunneled is used within a tunnel, the tunneling server can forward CHAP credentials to a legacy AAA server, such as a RADIUS server. No modification to the legacy AAA server is required.

1.2 Using EAP-MD5-Tunneled to Secure EAP-MD5-Challenge

EAP-MD5-Tunneled can easily be converted to CHAP because the CHAP protocol, as used in RADIUS, allows an intermediary - in this case, the tunneling server - to generate the challenge.

However, CHAP's alter ego EAP-MD5-Challenge does not permit this; instead, the server terminating the EAP-MD5-Challenge protocol is responsible for generating challenge material, to ensure freshness.

Therefore, in cases where it is desirable to forward an EAP-MD5-Challenge to a home authentication server, the collusion of that server is required. By properly constructing its challenge, the home authentication server can allow the tunneling server to perform EAP-MD5-Tunneled inside the tunnel and convert it to EAP-MD5-Challenge outside the tunnel.

Note that no alteration of the EAP-MD5-Challenge protocol is required to do this. The home server would need to follow certain rules when creating its challenge; however, it would remain interoperable with any client using ordinary EAP-MD5-Challenge outside a tunnel.

Unlike the CHAP case, however, home servers would require modification in order to perform EAP-MD5-Challenge in a manner compatible with EAP-MD5-Tunneled.

2. Architectural Model

The network architectural model for EAP-MD5-Tunneled usage is shown below.

The entities depicted are logical entities and may or may not correspond to separate network components. For example, the Tunnel Server and Home Server may be the same device. Entities that may be required in practice but are not relevant to this discussion, such access points or AAA proxies, are not shown.

```
+--------+                   +--------+                   +--------+
|        | EAP-MD5-Tunneled  |        |       CHAP        |        |
| Client |<---- within ----->| Tunnel |<------ or ------->|  Home  |
|        |    EAP tunnel     | Server | EAP-MD5-Challenge | Server |
|        |                   |        |                   |        |
+--------+                   +--------+                   +--------+
```

- The Client is a device seeking access to the network.

- The Tunnel Server is a AAA device, such as a RADIUS server, that is trusted by the client and is able to terminate an EAP tunneling protocol such as EAP-TTLS or EAP-PEAP, once it has proven its identity via certificate or other credentials.

- The Home Server is a AAA device, such as a RADIUS server, that is able to authenticate users via CHAP or EAP-MD5-Challenge.

The Client and Tunnel Server negotiate an EAP tunneling protocol such as EAP-TTLS or EAP-PEAP. Within the tunnel, they perform EAP-MD5-Tunneled as an inner authentication protocol.

The authentication of the client is performed on the Home Server via CHAP or EAP-MD5-Challenge.

The Tunnel Server transforms EAP-MD5-Tunneled data within the tunnel to CHAP or EAP-MD5-Challenge data outside the tunnel when forwarding to the Home Server. This transformation is one-way; untunneled CHAP or EAP-MD5-Challenge data received by an attacker cannot be transformed into EAP-MD5-Tunneled data for use inside a tunnel.

3. Algorithm Overview

The basic idea behind EAP-MD5-Tunneled is that the MD5 digest operation is split between the Client and Tunnel Server. The Client responds to a challenge, not with the complete MD5 digest of password plus challenge, but rather with an intermediate result of the MD5 algorithm. The Tunnel Server receives the intermediate result and completes the MD5 algorithm. The Tunnel Server then has a challenge and response which are identical to that produced by normal CHAP; this challenge and response can be forwarded to the Home Server -- for example, a legacy RADIUS server -- as a normal CHAP or EAP-MD5-Challenge authentication.

An attacker can no longer dupe a client using untunneled CHAP or EAP-MD5-Challenge, because any credentials sent by that client are useless within the EAP-MD5-Tunneled protocol.

4. A Review of MD5

Because EAP-MD5-Tunneled does not use MD5 as a black box algorithm, but instead gets into the middle of it, a brief review of the pertinent features of the MD5 algorithm is provided here.

MD5 is an iterative algorithm that applies a hashing function repeatedly to successive 64-octet blocks of a message until the entire message has been hashed. The message may be of any length. Prior to applying the hashing function, padding is appended to the message, consisting of a single octet of value 80 hex, 0 or more octets of value 0 to bring the message length to a multiple of 64 octets less 8 octets, and 8 octets indicating the length, in bits,

of the original message. Thus the total length of the input to the hashing algorithm is a multiple of 64 octets.

Each iteration of the algorithm applies the hashing function to two parameters: a 16-octet vector, and a 64-octet segment of the padded message.

Let f(x, y) be the hashing function, V[n] be the nth 16-octet vector, and M[n] be the nth 64-octet message block (where n is 0-based). Then,

```
V[0] = I, the fixed initialization vector defined for MD5
V[n + 1] = f(V[n], M[n])
```

The final output of the MD5 algorithm is the 16-octet vector given by V[N], where N is the number of 64-octet blocks in the padded message.

5. The Algorithm, in Pictures

The diagrams below illustrate how the hashing function is applied in both standard MD5 processing and the modified processing used in EAP-MD5-Challenge. For simplicity, it is assumed that the total padded message length is 128 octets, so two applications of the hashing function are required. These diagrams are for explication only; a formal description of the algorithm follows.

The following diagram illustrates the MD5 processing when using standard CHAP or EAP-MD5-Challenge.

```
                    <------------------ 128 octets ------------------>
   +-------------+  +------------------------------------------------+
   |  V[0] = I   |  |ID| Password |  CHAP Challenge  |    padding    |
   +-------------+  +------------------------------------------------+
         |           \_________   __________/\__________   _________/
         v                     \ /                      \ /
   +-------------+              |                        |
   |  hashing    |              |                        |
   |  function   |<-------------+                        |
   |  (client)   |                                       |
   +-------------+                                       |
         |                                               |
         v                                               |
   +-------------+                                       |
   |    V[1]     |                                       |
   +-------------+                                       |
         |                                               |
         v                                               |
   +-------------+                                       |
   |  hashing    |                                       |
   |  function   |<--------------------------------------+
   |  (client)   |
   +-------------+
         |
         v
   +-------------+
   |CHAP Response|--------- to server -------------------------->
   +-------------+
```

The following diagram illustrates the MD5 processing when using EAP-MD5-Tunneled. Note that the Client sends an intermediate result to the Tunnel Server, which completes the MD computation.

```
                    <------------------ 128 octets ------------------>
   +-------------+  +------------------------------------------------+
   |  V[0] = I   |  |ID| Password |  CHAP Challenge  |    padding    |
   +-------------+  +------------------------------------------------+
         |           \__________  __________/\__________   _________/
         v                     \ /                      \ /
   +-------------+              |                        |
   |  hashing    |              |                        |
   |  function   |<-------------+                        |
   |  (client)   |                                       |
   +-------------+                                       |
         |                                               |
         v                                               |
   +-------------+                                       |
   |   V[1]      |                                       |
   +-------------+                                       v
         |                                        +-------------+
         |                                        |  hashing    |
         +----------------- to server ----------->|  function   |
                                                  |  (server)   |
                                                  +-------------+
                                                         |
                                                         v
                                                  +-------------+
                                                  |CHAP Response|
                                                  +-------------+
```

6. The Algorithm, in Formal Notation

The following terms are used in the algorithm specification below. Note that vertical bar "|" indicates concatenation.

- f(x, y) is the MD5 hashing function, where x is a 16-octet vector and y is a 64-octet message segment.

- V is a 16-octet vector that is input to and output by the MD5 hashing function, where:

  V[0] = I, the fixed MD5 initialization vector that is input to the first iteration of the hashing function f(x, y).

  V[n], for n > 0, is the output of the (n - 1)th, and input to the nth, iteration of the hashing function f(x, y).

- L(x) is the length, in octets, of a sequence x.

- ID is the one octet EAP identifier.

- P is the user password.

- C is the challenge.

- S is the sequence ID | P | C, which is the entire sequence to be hashed via MD5.

- C1 and C2 are two subsequences of the challenge C, which, when concatenated, form the entire challenge; that is, C = C1 | C2. C1 is the portion of the challenge that will be hashed by the client; C2 is the portion that will be added to the hash by the server. The challenge is partitioned such that L(ID | P | C1) is an exact multiple of 64 octets, as described more fully below.

- C1MIN is the minimum length of C1 that is required to provide sufficient challenge entropy; thus, L(C1) must be >= C1MIN.

- S' is the sequence ID | P | C1, which is the portion of S which is hashed by the client. L(S') will be a multiple of 64.

- N' is the number of 64-octet segments in S'; that is, L(S') = 64 * N'.

- R' is the client's 16-octet response, based on the modified use of the MD5 algorithm.

- R is the response computed by the server, by extending R' with the remaining challenge material. Thus, R is the response that would result by performing a normal CHAP or EAP-MD5-Challenge with the complete challenge C.

6.1 Server Issues Challenge

For ordinary CHAP or EAP-MD5-Challenge, an authentication server or NAS generates a random challenge of a length equal to the amount of entropy desired; typically this is 16 octets.

In EAP-MD5-Tunneled, an additional 63 octets of challenge material beyond the minimum desired entropy C1MIN is generated. Depending on the password length, anywhere from 0 to 63 octets of the additional challenge will actually be used by the client as input to MD5. Thus,

```
L(C) = C1MIN + 63
```

Note that there is a conditioning rule for generating challenges that is required to prevent a coincidence that, though unlikely, would fatally undermine the security of this protocol if it occurred. This rule is discussed below.

6.2 Client Computes Response

The Client receives the challenge C, and partitions it into C1 and C2, such that the length of the sequence S' is the largest possible multiple of 64. That is,

```
   L(C2) = L(ID | P | C) mod 64
```

Now, following normal CHAP/EAP-MD5-Challenge practice, the Client concatenates CHAP identifier, password and the C1 portion of the challenge to form the sequence S':

```
   S' = ID | P | C1
```

The Client now has a sequence for input to the MD5 algorithm, whose length is an exact multiple of 64 octets; that is, N' * 64.

The Client passes this sequence to the MD5 algorithm, and performs N' iterations of the hashing function f(x, y).

Note that MD5 padding would normally add an additional 64 octets of padding to this sequence - an octet of 80 hex, 55 octets of 0 and 8 octets of length; however, in the Client's modified use of the MD5 algorithm, this padding will not be input to the MD5 hashing function.

The Client response is computed using iterations of the MD5 hashing function f(x, y), as described above, where the final response R' is given by:

```
   R' = V[N']
```

To simply sum up this process, the Client truncates the challenge to cause the input to the MD5 algorithm be an exact multiple of 64 octets, then performs all MD5 iterations except the final one to produce its response.

6.3 Client Sends Response To Tunnel Server

The Client sends its response R' to the Tunnel Server. In addition, the Client sends the number of octets in the password L(P); this is necessary to allow the Tunnel Server to complete the MD5 digest.

6.4 Tunnel Server Processes Response

Having received R' and L(P), the Tunnel Server can now compute the correct response to the original challenge, by first recreating the state of the MD5 algorithm at the point that the Client left off, then completing the algorithm.

The state of the algorithm after an iteration of the hashing function consists of the count of octets already processed, and the value of the vector V.

The Tunnel Server can compute the count state variable based on the password size reported by the Client; the vector V is just R'.

Thus, the steps to complete the MD5 algorithm are as follows:

1 Based on the password length L(P) reported by the Client, compute the length of C2:

   L(C2) = (L(ID) + L(P) + L(C)) mod 64

2 Compute the length of the sequence already hashed by the Client:

   L(S') = (L(ID) + L(P) + L(C)) - L(C2)

3 Initialize the MD5 algorithm to the state at which it was left by the Client, by setting the initialization vector to R' and the count to L(S').

4 Compute R, the response that ordinary CHAP/EAP-MD5-Challenge would yield, by completing the MD5 algorithm using C2 as message input, and applying normal padding. C2 will be the last L(C2) octets of the challenge, which may be anywhere from 0 through 63 octets.

6.5 Tunnel Server Validates Response Locally or Remotely

If the Tunnel Server knows the user's password (that is, it is also the Home Server), it can validate the Client response directly, simply by concatenating the EAP identifier ID, the user's known password P, and the full challenge C, performing an MD5 digest on that sequence, and comparing the result to R.

If the Tunnel Server does not know the user's password, it may forward CHAP or EAP-MD5-Challenge values to the Home Server via RADIUS proxy or other means. These values will include ID, C and R.

6.6 Challenge Conditioning

There is a precaution that a Tunnel Server must take when issuing a challenge, in order to eliminate the possibility that the modified version of MD5 used by this protocol can produce a result that is an alias of an actual MD5 digest.

The last 8 octets of MD5 padding are set to the length of the message in bits. It is possible, with a random challenge, that the last 8 octets of the portion of the challenge used by the Client (C1) can be within a numeric range that would allow a man-in-the-middle attacker to formulate a challenge to a Client using non-tunneled EAP-MD5-Challenge and elicit a response which is identical to the response that would result from EAP-MD5-Tunneled. This would allow precisely the attack that this protocol is designed to avoid.

The Tunnel Server must condition the challenge to ensure that no sequence of 8 octets can possibly alias a bit length that can naturally occur, by observing the following rule:

- Each octet in the challenge must be non-0.

This can easily be done, for example, by generating a random challenge and converting all octets of value 0 to octets of value 1. Note that the amount by which this reduces the entropy of the challenge is insignificant.

The absence of octets of value 0 in the challenge ensures that any sequence of 8 octets will indicate an impossibly high bit length.

The Client, for its part, must refuse to proceed with authentication if it receives a challenge that does not meet the above condition.

7. Using EAP-MD5-Tunneled with EAP-MD5-Challenge

In order for Tunnel Server to use EAP-MD5-Tunneled within a tunnel to the Client, while proxying an EAP-MD5-Challenge outside the tunnel to a Home Server, the Home Server must follow these rules:

- The Home Server must provide at least 79 octets of challenge material. This implies that C1MIN is always at least 16 octets, which is generally considered sufficient challenge entropy.

- The Home Server must condition the challenge as described above; that is, it must not contain any octets of value 0.

When the Tunnel Server receives the initial EAP-MD5-Challenge request from the Home Server, it simply uses that challenge rather than issue its own, as it would if it were forwarding CHAP.

The Tunnel Server must require that the challenge received from the Home Server be at least 79 octets and not contain 0 octets; if these conditions are not met, it must not perform an EAP-MD5-Tunneled authentication with the Client.

8. Packet Formats

EAP-MD5-Tunneled is performed in a single EAP round trip; that is, a single EAP Request from the server and a single EAP Response from the client.

The packet formats below show EAP payloads only, not the EAP header.

8.1 EAP Request

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Type      |  Value-Size   |        Challenge ...          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
   |                                                               |
   .                                                               .
   .                                                               .
   .                                                               .
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Name ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type

This field is one octet and is set to (tbd).

Value-Size

This field is one octet and indicates the length of the Challenge field.

Challenge

This field contains the challenge value C, as described above. It is a variable length random sequence of octets, whose length is C1MIN + 63, where C1MIN is determined by server policy, and is typically 16 octets or more. The challenge value must be conditioned by ensuring the absence of any octets of value 0, as described above.

Name

This field is of variable length and may be omitted entirely. It may be used to identify the system transmitting the packet. There are no limitations on the content of this field. For example, it MAY contain ASCII character strings or globally unique identifiers in ASN.1 syntax. The Name should not be NUL or CR/LF terminated. Its size is inferred from the Length field of the EAP header.

8.2 EAP 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
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |     Type      |  Value-Size   |        Response ...           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
    |                                                               |
    +                                                               +
    |                                                               |
    +                                                               +
    |                                                               |
    +                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                               |       Password-Length         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Name ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type

This field is one octet and is set to (tbd).

Value-Size

This field is one octet and indicates the length of the Response field. It is always 16.

Response

This field is 16 octets and contains the response value R', as described above.

Password-Length

This field is two octets and indicates the length, in octets, of the user's password L(P).

Name

This field is of variable length and may be omitted entirely. It may be used to identify the system transmitting the packet. There are no limitations on the content of this field. For example, it MAY contain ASCII character strings or globally unique identifiers in ASN.1 syntax. The Name should not be NUL or CR/LF terminated. Its size is inferred from the Length field of the EAP header.

9. Security Considerations

9.1 Security Policy Constraints

The purpose of EAP-MD5-Tunneled is to provide an authentication protocol that is different from protocols already in use, in order to prevent a man-in-the-middle attack that depends on the same protocol with the same credentials being used both inside and outside a tunnel.

Therefore, EAP-MD5-Tunneled MUST NOT be used except within a tunneled EAP protocol such as EAP-TTLS or EAP-PEAP. The tunneling protocol must at a minimum provide for encryption of the inner EAP-MD5-Tunneled data and provide for strong trust of the server by the client via certificate or other means.

Both Client and Tunnel Server must adhere to the above constraint.

A Tunnel Server can completely protect against the attack against tunneled protocols by refusing to perform CHAP or EAP-MD5-Challenge within a tunnel, and by only permitting EAP-MD5-Tunneled to be used for CHAP-type credentials.

9.2 Revelation of Password Length

Note that the Client must indicate to the Tunnel Server the length of the user's password. This reduces somewhat the entropy of the password, as seen by the Tunnel Server, and would make a dictionary attack slightly easier to mount.

This length, however, is only revealed to the Tunnel Server. Because the information is in an encrypted tunnel, it is not available to eavesdroppers. Also, because the challenge that is forwarded to the Home Server is of independent length, the password length is not revealed when a CHAP or EAP-MD5-Challenge authentication is forwarded outside the tunnel.

This is not felt to be a security concern, because the server that is the tunnel endpoint is explicitly trusted by the user, and trusted in particular not to mount dictionary attacks when using challenge/response protocols based on passwords.

9.2.1 Relation of Entropy to Feasibility of Dictionary Attack

It is also worth noting that the decrease in resistance to dictionary attack is less than the reduction in entropy would seem to indicate. A dictionary attack proceeds by brute force, examining candidate passwords in decreasing order of likelihood. Since longer passwords are usually less likely than shorter ones, a dictionary attack will generally examine shorter passwords before longer ones. Knowledge of password length, therefore, serves to eliminate from

consideration mostly candidate passwords that are shorter than the known length, rather than longer. Since the number of possible passwords increases with size, elimination of shorter passwords represents a relatively small population of eliminated passwords compared to the number of passwords of the actual length that must be examined.

To understand why this is so, consider an extreme case in which there are 8 possible password characters, all equally probable; and all password lengths are also equally probable over a certain range. With such a password pool, the likelihood of a particular password is strictly related to its length; that is, all 5-character passwords are more likely that 6-character passwords. Therefore, a dictionary attack would examine all 5-character passwords before continuing on to 6-character passwords, etc. If it is known that the password is 6 characters, the attacker only saves having to examine passwords of 5 characters or fewer; passwords of 7 characters or more would not have been examined in any case, since examination of the 6-character passwords would already have yielded a result. But the number of 5-character passwords is 1/8 of the number of 6-character passwords, the number of 4-character passwords is 1/64, etc. So the number of candidate passwords eliminated from consideration is quite small relative to the number that would still have to be examined.

Of course, in reality the password characters will not be equi-probable, and the variance in probability means that, for example, there will be some 7-character passwords that are more likely than 6-character passwords. Still, the general correlation of length to likelihood holds, and if knowledge of password length represents a entropy reduction of, say, 4 bits, the reduction of work for a dictionary attack is considerably less than a factor of 16.

10. References

[MD5] Rivest, R., and Dusse, S., "The MD5 Message-Digest Algorithm", MIT Laboratory for Computer Science and RSA Data Security, Inc., RFC 1321, April 1992.

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

[PPP] Simpson, W., Editor, "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, July 1994.

[CHAP] Simpson, W., "PPP Challenge Handshake Authentication Protocol (CHAP)", RFC 1994, August 1996.

[RADIUS] Rigney, C., Rubens, A., Simpson, W., and Willens, S., "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[TLS]	Dierks, T., and Allen, C., "The TLS Protocol Version 1.0", RFC 2246, November 1998.
[TTLS]	Funk, P., and Blake-Wilson, S., "EAP Tunneled TLS Authentication Protocol (EAP-TTLS)", draft-ietf-pppext-eap-ttls-02.txt, November 2002.
[PEAP]	Andersson, H., Josefsson, S., Zorn, G., Simon, D., and Palekar, A., "Protected EAP Protocol (PEAP)", draft-josefsson-pppext-eap-tls-eap-05.txt, September 2002.
[MITM]	Asokan, N., Niemi, V., and Nyberg, K., "Man-in-the-Middle in Tunneled Authentication", http://www.saunalahti.fi/~asokan/research/mitm.html, Nokia Research Center, Finland, October 24 2002.
[BINDING]	Puthenkulam, J., Lortz, V., Palekar, A., Simon, D., and Aboba, B., "The Compound Authentication Binding Problem", draft-puthenkulam-eap-binding-01.txt, October 2002.
[NUMBERS]	Reynolds, J., and Postel, J., "Assigned Numbers", RFC 1700, October 1994.

11. Author's Address

Questions about this memo can be directed to:

Paul Funk
Funk Software, Inc.
222 Third Street
Cambridge, MA 02142
USA

Phone: +1 617 497-6339
E-mail: paul@funk.com

12. Intellectual Property Rights Notice

The IETF has been notified of intellectual property rights claimed in regard to some or all of the specification contained in this document. For more information consult the online list of claimed rights.

13. Full Copyright Statement