RADIUS Working Group
INTERNET-DRAFT
<draft-ietf-radius-saltencrypt-00.txt>

P. Funk
O. Tavakoli
Funk Software, Inc.
D. Mitton
D. Fox
Bay Networks, Inc.
November 20, 1997

Salt-Encryption of RADIUS Attributes

1. Status of this Memo

2. Abstract

This document defines a general mechanism for encrypting attributes within RADIUS packets. This mechanism permits more than one attribute within a RADIUS transaction (request and response) to be encrypted without compromising the security of the encryption.

3. Introduction

For security reasons, it is necessary to encrypt certain attributes that are passed between a NAS and a RADIUS server.

RADIUS [1] defines a password-hiding mechanism for use with the User-Password attribute in an Access-Request; namely, that the Value of the attribute is XORed with an octet sequence based on a one-way MD5 digest of the shared secret and the Request Authenticator.

This mechanism is not extensible to additional attributes in the request packet or the RADIUS server s response packet without compromising the encryption. This is because the first 16 octets of the XOR value will be identical for each encryption, allowing an

attacker who knows the clear text value of any of the encrypted

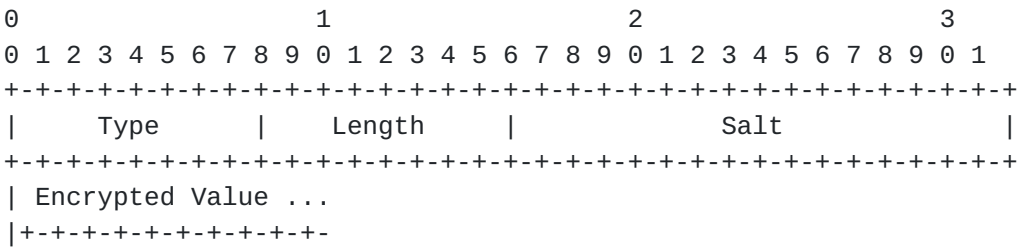
attributes to deduce the common XOR value and decipher the other encrypted attributes.

The mechanism defined here -- called "salt-encryption" -- adds a unique two-octet Salt value to each attribute to be encrypted. This Salt would be concatenated with the shared secret and Request Authenticator as input to the MD5 digest to produce an initial 16-byte XOR value that is unique for each encrypted attribute in a RADIUS transaction. The initial and subsequent XOR values are used to encrypt the payload of the attribute. The length of the actual information portion of the attribute MAY be obfuscated by encoding the payload with the length of the actual data, followed by the data, followed by optional padding.

4. Attribute Format

4.1 Standard Form

A summary of the standard form of the salt-encrypted attribute format is shown below. The fields are transmitted from left to right.



Type

The Type field is a single octet as defined in [1].

Length

The Length field is a single octet as defined in [1].

Salt

The Salt field is two octets, and is used to differentiate encryption keys that are based on the same shared secret and Request Authenticator.

The NAS and the RADIUS server are responsible for ensuring that each salt within a single packet is unique. To ensure uniqueness across a pair of packets constituting a transaction, each Salt in an Access-Request packet sent by the NAS must have high-bit clear, and each Salt in an Access-Accept, Access-Reject, or Access-Challenge packet returned by the RADIUS server must have high-bit set.

Encrypted Value

The Encrypted Value field is one or more octets, encrypted according to the mechanism described below, containing data that is length-prefixed and optionally padded.

The first octet indicates the number of significant data octets to follow, excluding any padding.

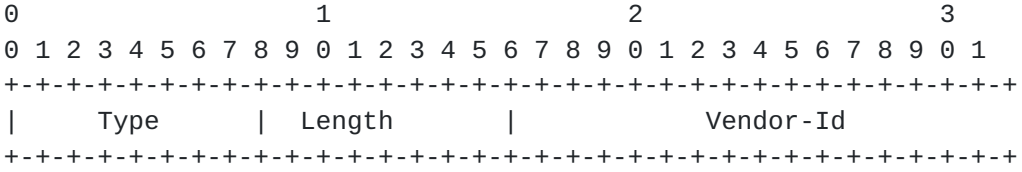
The data that follows the first octet contains the information specific to the Attribute.

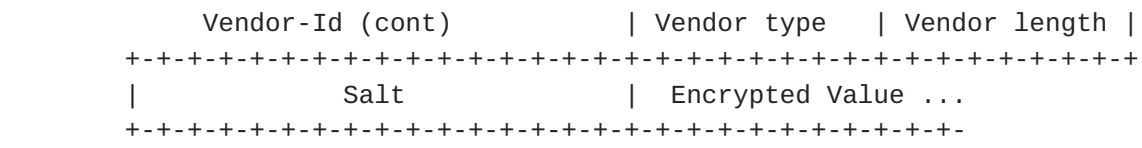
Following the data, there may be additional octets of padding that carry no information but serve to obfuscate the actual length of the data. The technique used may be null-padding up to the next multiple of 16 octets (as in the password-hiding mechanism defined in [1]), padding by a random number of octets, or some other method.

4.2 Standard Form for Vendor-Specific Attributes

A Vendor-Specific attribute consists of an attribute of type 26 that contains a Vendor-Id and vendor-defined information. According to [1], the vendor-defined information SHOULD consist of a sequence of one or more sub-attributes, each of which consists of a Vendor type and Vendor length.

A sub-attribute of a Vendor-Specific attribute may be salt-encrypted using a format corresponding to an ordinary attribute. A summary of the salt-encrypted Vendor-Specific Attribute format is shown below. The fields are transmitted from left to right.





All fields are as defined in [1] or as defined above.

4.3 Alternative Forms

New attributes may be defined that utilize salt-encryption without strictly adhering to the standard formats described above. For example, it might be desirable to encrypt only part of an attribute, keeping the rest in clear text; or to include multiple salts within an attribute to encrypt multiple sub-fields. It may also be desirable

to eliminate the length prefix and padding from the Encrypted Value, particularly for fixed-length data where length obfuscation provides no benefit.

The exact arrangement of the Salt field and the Encrypted Value field within an attribute, and whether the Encrypted Value field utilizes length-obfuscation, is a matter to be decided for each new attribute as it is defined. It is expected that some, but not all, new attributes will follow the standard formats as described above.

All salt-encrypted attributes MUST at least observe the following requirements: Each Salt is 2 octets, unique within the packet, with high-bit clear in requests and set in responses, and the encryption/decryption follows the method outlined below.

5. Method of encryption/decryption

The salt-encryption method closely corresponds the password-hiding method defined in [1]. The differences are:

(1) The Salt is concatenated to the shared secret and Request Authenticator when computing the initial MD5 digest.

(2) An attribute may be padded to an arbitrary length or not at all. However, in order to obfuscate the actual length of the data, a padding strategy, such as null-padding to a multiple of 16 octets, SHOULD be employed.

The salt-encryption method proceeds as follows:

Construct a clear text version of the information to be encrypted;

call this the Clear Text.

Call the shared secret S, the pseudo-random 128-bit Request Authenticator RA, and the Salt SALT. Break the Clear Text into chunks p1, p2, etc. of up to 16-octets each; the last chunk may contain fewer than 16 octets. Call the ciphertext blocks c(1), c(2), etc. We'll need intermediate values b1, b2, etc.

```
b1 = MD5(S + RA + SALT)    c(1) = p1 xor b1
b2 = MD5(S + c(1))         c(2) = p2 xor b2
                .                     .
                .                     .
                .                     .
bi = MD5(S + c(i-1))       c(i) = pi xor bi
```

Note that if the last chunk is fewer than 16 octets only the first part of the final MD5 digest bi is used in the XOR operation.

The resulting Encrypted Value will contain c(1)+c(2)+...+c(i) where + denotes concatenation.

On receipt, the process is reversed to yield the Clear Text.

6. Security Considerations

Security is the subject of this document.

7. Author s Addresses

Authors may be contacted by email as follows:

Paul Funk paul@funk.com
Oliver Tavakoli oliver@funk.com
Dave Mitton dmitton@baynetworks.com
Daniel Fox dfox@baynetworks.com

8. Expiration Date

This document expires May 25, 1998.

9. References

[1] Rigney, C., et. al., "Remote Access Dial In User Service", RFC 2138, April 1997

Paul Funk	Tel: +1 617 497 6339
Funk Software, Inc.	Fax: +1 617 547 1031
222 Third Street	Internet: paul@funk.com
Cambridge, MA 02142 USA	