SASL Working Group
Internet-Draft
Obsoletes: RFC2195
(if approved)
Intended status: Standards Track
Expires: January 12, 2009

L. Nerenberg, Ed.
Orthanc Systems

July 11, 2008

The CRAM-MD5 SASL Mechanism
draft-ietf-sasl-crammd5-10

Status of this Memo

Abstract

This document defines a simple challenge-response authentication mechanism, using a keyed MD5 digest, for use with the Simple Authentication and Security Layer (SASL).

Table of Contents

1. Introduction

This document defines a simple challenge-response authentication method, using a keyed MD5 [RFC2104] digest, for use with the Simple Security and Authentication Layer (SASL) [RFC4422]. The mechanism name associated with CRAM-MD5 is 'CRAM-MD5'.

This mechanism does not provide a security layer.

The CRAM-MD5 mechanism is intended to have limited use on the Internet. The mechanism offers inadequate protection against common attacks against application-level protocols (see Section 5) and is prone to interoperability problems (see Section 4).

2. The CRAM-MD5 SASL Mechanism

The mechanism starts with the server issuing a <challenge>. The data contained in the challenge contains a string of random data.

The client makes note of the data and then responds with a <response> consisting of the <username>, a space, and a <digest>. The digest is computed by applying the keyed MD5 algorithm from [RFC2104] where the key is a shared secret and the digested text is the <challenge> (including angle-brackets). The client MUST NOT interpret or attempt to validate the contents of the challenge in any way.

This shared secret is a string known only to the client and server. The digest parameter itself is a 16-octet value which is sent in a restricted hexadecimal format (see the <digest> production in Section 3).

When the server receives this client response, it verifies the digest provided. Since the user name may contain the space character, the server MUST ensure the right-most space character is recognised as the token separating the user name from the digest. If the digest is correct, the server should consider the client authenticated.

3. Formal Grammar

The following grammar specification uses the Augmented Backus-Naur Form (ABNF) as specified in [RFC4234], and incorporates by reference the Core Rules defined in that document.

```
challenge  = "<" 3*(%x21-3B / %x3D / %x3F-7E) ">"
             ; a bracketed string of printing ASCII characters, not
             ; containing embedded "<" or ">"

digest     = 32(DIGIT / %x61-66)
             ; A hexadecimal string, using ONLY lower-case
             ; letters

response   = username SP digest

username   = 1*OCTET
             ; SHOULD be well-formed UTF-8
```

4. Interoperability Considerations

The design of CRAM-MD5 [RFC2095] pre-dated any widespread use of UTF-8 to encode protocol elements. It was initially deployed as an extension to the IMAP4 protocol at a time when authentication and authorization identifiers were almost exclusively encoded in the US-ASCII character set, therefore it is silent about the encoding and representation of non-US-ASCII data elements. When sites first began using alternate character sets to encode user names (and passwords) they simply used the raw 8-bit character representation. This works - for the most part - but only because these enclaves tend to use a common character set amongst themselves. When a second group of users using a different character set is introduced into the mix, interoperability suffers.

So as not to render existing implementations non-compliant this update preserves the existing opaque nature of user names and passwords. However, implementors are strongly encouraged to process the user name and password data as described in the next paragraph. Doing so prevents interoperability problems caused by incompatible character set encodings.

The client SHOULD prepare the user name and shared secret strings using the SASLprep [RFC4013] profile of the Stringprep [RFC3454] algorithm. The resulting values SHOULD be encoded as UTF-8 [RFC3629] strings. The server may store the prepared string instead of, or as well as, the unprepared string, so that it does not have to prepare it every time it is needed for computation. However, if the original (unprepared) string is not stored, it may render the computed secret to be incompatible with a future revisions of SASLprep that support currently unassigned code points (see section 7 of [RFC3454]). It is therefor recommended to store the unprepared string in the database.

5. Security Considerations

CRAM-MD5 is no longer considered to provide adequate protection.

This mechanism is vulnerable to dictionary attack by any passive listener able to observe the user name, challenge and response. An attacker can use the user name and challenge to compute a series of responses based on a pass-phrase dictionary, looking for a match to the response sent by the client.

CRAM-MD5 does not authenticate the server and does not include a client-supplied nonce. Consequently, it is possible to construct a server with a fixed challenge string that has pre-computed the hashes for all possible passwords up to a certain length (or from a dictionary). Such a server could then immediately determine the user's password if it is sufficiently short or non-random.

This mechanism does not obscure the user name in any way. Accordingly, a server that implements both a clear-text password command and this authentication type should not allow both methods of access for a given user name.

For the reasons described above, CRAM-MD5 SHOULD NOT be used unless the application protocol session is protected by an encryption layer, such as provided by TLS.

Keyed MD5 is chosen for this application because of the greater security imparted to authentication of short messages. In addition, the use of the techniques described in [RFC2104] for pre-computation of intermediate results make it possible to avoid explicit clear-text storage of the shared secret on the server system by instead storing the intermediate results which are known as "contexts." While the saving, on the server, of the MD5 context is marginally better than saving the shared secrets in clear-text, it is not sufficient to protect the secrets if the server itself is compromised. Consequently, servers that store the secrets or contexts must both be protected to a level appropriate to the potential information value in the data and services protected by this mechanism. In other words, techniques like this one involve a trade-off between vulnerability to network sniffing and I/O buffer snooping and vulnerability of the server host's databases. If one believes that the host and its databases are subject to compromise, and the network is not, this technique (and all others like it) is unattractive. It is perhaps even less attractive than clear-text passwords, which are typically stored on hosts in one-way hash form. On the other hand, if the server databases are perceived as reasonably secure, and one is concerned about client-side or network interception of the passwords (secrets), then this (and similar) techniques are

preferable to clear-text passwords by a wide margin.

While there are now suggestions in the literature that the use of MD5 and keyed MD5 in authentication procedures probably has a limited effective lifetime, the technique is now widely deployed and widely understood. It is believed that this general understanding may assist with the rapid replacement, by CRAM-MD5, of the current uses of permanent clear-text passwords in many protocols. This document has been deliberately written to permit easy upgrading to use SHA (or whatever alternatives emerge) when they are considered to be widely available and adequately safe.

Even with the use of CRAM-MD5, users are still vulnerable to active attacks. An example of an increasingly common active attack is 'TCP Session Hijacking' as described in CERT Advisory CA-95:01.

6. References

6.1. Normative References

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC3454] Hoffman, P. and M. Blanchet, "Preparation of Internationalized Strings ("stringprep")", RFC 3454, December 2002.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC4013] Zeilenga, K., "SASLprep: Stringprep Profile for User Names and Passwords", RFC 4013, February 2005.

[RFC4234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.

[RFC4422] Melnikov, A. and K. Zeilenga, "Simple Authentication and Security Layer (SASL)", RFC 4422, June 2006.

6.2. Informative References

[RFC2095] Klensin, J., Catoe, R., and P. Krumviede, "IMAP/POP AUTHorize Extension for Simple Challenge/Response", RFC 2095, January 1997.

[RFC2244] Newman, C. and J. Myers, "ACAP -- Application

Configuration Access Protocol", RFC 2244, November 1997.

[RFC3501] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, March 2003.

[RFC4616] Zeilenga, K., "The PLAIN Simple Authentication and Security Layer (SASL) Mechanism", RFC 4616, August 2006.

Appendix A. Examples

The examples in this appendix DO NOT form part of the specification. Where conflicts exist between the examples and the formal grammar or the normative text in Section 2, the latter are authoritative.

A.1. IMAP4

These examples show the use of the CRAM-MD5 mechanism with the IMAP4 [RFC3501] AUTHENTICATE command. The base64 encoding of the challenges and responses is part of the IMAP4 AUTHENTICATE command, and not part of the CRAM-MD5 specification itself.

A.1.1. Example 1: Simple IMAP

In this example the shared secret is the string 'tanstaaftanstaaf'.

```
S: * OK [CAPABILITY IMAP4rev1 STARTTLS LOGINDISABLED AUTH=CRAM-MD5]
C: A0001 AUTHENTICATE CRAM-MD5
S: + PDE4OTYuNjk3MTcwOTUyQHBvc3RvZmZpY2UuZXhhbXBsZS5uZXQ+
C: am9lIDNkYmM4OGYwNjI0Nzc2YTczN2IzOTA5M2Y2ZWI2NDI3
S: A0001 OK CRAM-MD5 authentication successful
```

Hence, the keyed MD5 digest is produced by calculating

```
MD5((SASLprep(tanstaaftanstaaf) XOR opad),
    MD5((SASLprep(tanstaaftanstaaf) XOR ipad),
       <1896.697170952@postoffice.example.net>))
```

where ipad and opad are as defined in RFC 2104 and the string shown in the challenge is the base64 encoding of '<1896.697170952@postoffice.example.net>'. The shared secret is null-padded to a length of 64 bytes. If the shared secret is longer than 64 bytes, the MD5 digest of the shared secret is used as a 16 byte input to the keyed MD5 calculation.

This produces a digest value (in hexadecimal) of '3dbc88f0624776a737b39093f6eb6427'. The user name is then prepended

to it, forming 'joe 3dbc88f0624776a737b39093f6eb6427', which is then base64 encoded to meet the requirements of the IMAP4 AUTHENTICATE command yielding 'am9lIDNkYmM4OGYwNjI0Nzc2YTczN2IzOTA5M2Y2ZWI2NDI3'.

A.1.2. Example 2: IMAP4 with embedded spaces

This example uses the user name 'Ali Baba' and the shared secret 'Open, Sesame'. It illustrates that both user names and passwords may contain non-alphanumeric characters.

```
S: <68451038525716401353.0@localhost>
C: Ali Baba 6fa32b6e768f073132588e3418e00f71
```

A.1.3. Example 3: IMAP4 with Unicode characters

This example demonstrates the processing of Unicode strings. The raw user name is 'Al<U+00AA>dd<U+00AD>in<U+00AE>' where <U+00AA> is the Unicode Latin symbol <FEMININE ORDINAL INDICATOR>, <U+00AD> is <SOFT HYPHEN>, and <U+00AE> is the <REGISTERED SIGN>. Preparing the raw user name with SASLprep returns 'Aladdin<U+00AE>' which we then encode into the UTF-8 string 'Aladdin\xC2\xAE' (shown here and below using C-style string format notation). As before, the shared secret is 'Open, Sesame'.

```
S: <92230559549732219941.0@localhost>
C: Aladdin\xC2\xAE 9950ea407844a71e2f0cd3284cbd912d
```

A.2. ACAP

An example of using CRAM-MD5 with ACAP [RFC2244].

A.2.1. Example 4: Simple ACAP

This example uses the user name 'joe' and the shared secret 'tanstaaftanstaaf'.

```
S: * ACAP (IMPLEMENTATION "Infotrope ACAP Server, version 0.1.3,
    Copyright 2002-2004 Dave Cridland <dave@cridland.net>")
    (SASL "PLAIN" "DIGEST-MD5" "CRAM-MD5" "ANONYMOUS") (STARTTLS)
C: AUTH AUTHENTICATE "CRAM-MD5"
S: + {43}
S: <2262304172.6455022@gw2.gestalt.entity.net>
C: {36+}
C: joe 2aa383bf320a941d8209a7001ef6aeb6
S: AUTH OK "You're logged in as joe. Frooby."
```

Appendix B. IANA Considerations

It is requested that the Internet Assigned Numbers Authority (IANA) update the SASL Mechanism Registry entry for CRAM-MD5 to refer to this document.

To: iana@iana.org
Subject: Updated Registration of SASL CRAM-MD5 mechanism.

SASL mechanism name: CRAM-MD5
Security considerations: See RFC XXXX
Published specification: RFC XXXX
Person & email address to contact for further information:
Lyndon Nerenberg <lyndon+rfc-crammd5@orthanc.ca>
IETF SASL WG <ietf-sasl@imc.org>

Appendix C. Contributors

The CRAM-MD5 mechanism was originally specified in RFC 2095, IMAP/POP AUTHorize Extension for Simple Challenge/Response. The authors of that document -- John C. Klensin, Paul Krumviede, and Randy Catoe -- are to be credited with the design and specification of CRAM-MD5, and they are the original authors of the majority of the text in this document. This memo serves only to re-state CRAM-MD5 within the formal context of SASL, which specification it preceded by several months.

Dave Cridland and Simon Josefsson contributed updated examples.

Appendix D. Changes since RFC 2195

The syntax of the <challenge> has been relaxed.

A section on interoperability concerns has been added.

The security considerations have been updated to reflect the current views of the security community.

Author's Address

Lyndon Nerenberg (editor)
Orthanc Systems

Email: lyndon+rfc-crammd5@orthanc.ca

Full Copyright Statement

Intellectual Property