INTERNET-DRAFT
Intended Category: Standards Track
Expires December 2006
Updates: RFC 2595

Editor: Kurt D. Zeilenga
OpenLDAP Foundation

16 June 2006

The PLAIN SASL Mechanism
<draft-ietf-sasl-plain-09.txt>

Status of Memo

This document is intended to be, after appropriate review and revision, submitted to the RFC Editor as a Standards Track document. Distribution of this memo is unlimited. Technical discussion of this document will take place on the IETF SASL mailing list <ietf-sasl@imc.org>. Please send editorial comments directly to the document editor <Kurt@OpenLDAP.org>.

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

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.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Copyright (C) The Internet Society (2006). All Rights Reserved.

Please see the Full Copyright section near the end of this document for more information.

Abstract

This document defines a simple clear-text user/password Simple Authentication and Security Layer (SASL) mechanism called the PLAIN mechanism. The PLAIN mechanism is intended to be used, in combination with data confidentiality services provided by a lower layer, in protocols which lack a simple password authentication command.

1. Background and Intended Usage

Clear-text, multiple-use passwords are simple, interoperate with almost all existing operating system authentication databases, and are useful for a smooth transition to a more secure password-based authentication mechanism. The drawback is that they are unacceptable for use over network connection where data confidentiality is not assured (by encryption or other means).

This document defines the PLAIN Simple Authentication and Security Layer ([SASL]) mechanism for use in protocols with no clear-text login command (e.g., [ACAP] or [SMTP-AUTH]). This document updates RFC 2595, replacing Section 6. Changes since RFC 2595 are detailed in Appendix A.

The name associated with this mechanism is "PLAIN".

The PLAIN SASL mechanism does not provide a security layer.

The PLAIN mechanism should not be used without adequate data security protection as this mechanism affords no integrity nor confidentiality protections itself. The mechanism is intended to be used with data security protections provided by application layer protocol, generally through its use of Transport Layer Security ([TLS]) services.

By default, implementations SHOULD advertise and make use of the PLAIN mechanism only when adequate data security services are in place. Specifications for IETF protocols which indicate that this mechanism is an applicable authentication mechanism MUST mandate that implementations support an strong data security service, such as TLS.

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 [Keywords].

2. PLAIN SASL mechanism

The mechanism consists of a single message, a string of [UTF-8] encoded [Unicode] characters, from the client to the server. The

client presents the authorization identity (identity to act as), followed by a NULL (U+0000) character, followed by the authentication identity (identity whose password will be used), followed by a NULL (U+0000) character, followed by the clear-text password. As with other SASL mechanisms, the client does not provide an authorization identity when it wishes the server to derive an identity from the credentials and use that as the authorization identity.

The formal grammar for the client message using Augmented BNF [ABNF] follows.

```
message   = [authzid] UTF8NULL authcid UTF8NULL passwd
authcid   = 1*SAFE ; MUST accept up to 255 octets
authzid   = 1*SAFE ; MUST accept up to 255 octets
passwd    = 1*SAFE ; MUST accept up to 255 octets
UTF8NULL  = %x00 ; UTF-8 encoded NULL character

SAFE      = UTF1 / UTF2 / UTF3 / UTF4
            ;; any UTF-8 encoded Unicode character except NULL

UTF1      = %x01-7F ;; except NULL
UTF2      = %xC2-DF UTF0
UTF3      = %xE0 %xA0-BF UTF0 / %xE1-EC 2(UTF0) /
            %xED %x80-9F UTF0 / %xEE-EF 2(UTF0)
UTF4      = %xF0 %x90-BF 2(UTF0) / %xF1-F3 3(UTF0) /
            %xF4 %x80-8F 2(UTF0)
UTF0      = %x80-BF
```

The authorization identity (authzid), authentication identity (authcid), password (passwd), and NULL character deliminators SHALL be transferred as [UTF-8] encoded strings of [Unicode] characters. As the NULL (U+0000) character is used as a deliminator, the NULL (U+0000) character MUST NOT appear in authzid, authcid, or passwd productions.

The form of the authzid production is specific to the application-level protocol's SASL profile [SASL]. The authcid and passwd productions are form-free. Use of non-visible characters or characters which a user may be unable to enter on some keyboards is discouraged.

Servers MUST be capable of accepting authzid, authcid, and passwd productions up to and including 255 octets. It is noted that the UTF-8 encoding of a Unicode character may be as long as 4 octets.

Upon receipt of the message, the server will verify the presented (in the message) authentication identity (authcid) and password (passwd) with the system authentication database, and verify the authentication

credentials permit the client to act as the (presented or derived) authorization identity. If both steps succeed, the user is authenticated.

The presented authentication identity and password strings, as well as the database authentication identity and password strings, are to be prepared before being used in the verification process. The [SASLPrep] profile of the [StringPrep] algorithm is the RECOMMENDED preparation algorithm. The SASLprep preparation algorithm is recommended to improve the likelihood that comparisons behave in an expected manner. The SASLprep preparation algorithm is not so as to allow the server to employ other preparation algorithms (including none) when appropriate. For instance, use of different preparation algorithm may be necessary for the server to interoperate with an external system.

When preparing the presented strings using [SASLPrep], the presented strings are to be treated as "query" strings [Section 7, Stringprep] and hence unassigned code points are allowed appear in their prepared output. When preparing the database strings using [SASLprep], the database strings are to be treated as "stored" strings [Section 7, Stringprep] and hence unassigned code points are prohibited from appearing in their prepared output.

Regardless of the preparation algorithm used, if the output of a non-invertible function (e.g., hash) of the expected string is stored, the string MUST be prepared before input to that function.

Regardless of the preparation algorithm used, if preparation fails or results in an empty string, verification SHALL fail.

When no authorization identity is provided, the server derives an authorization identity from the prepared representation of the provided authentication identity string. This ensures that the derivation of different representations of the authentication identity produce the same authorization identity.

The server MAY use the credentials to initialize any new authentication database, such as one suitable for [CRAM-MD5] or [DIGEST-MD5].

4. Pseudo-Code

This section provides pseudo-code illustrating the verification process (using hashed passwords and the SASLprep preparation function) discussed above. This section is not definitive.

```
boolean Verify(string authzid, string authcid, string passwd) {
  string pAuthcid = SASLprep(authcid, true); # prepare authcid
  string pPasswd = SASLprep(passwd, true);   # prepare passwd
  if (pAuthcid == NULL || pPasswd == NULL) {
    return false;     # preparation failed
  }
  if (pAuthcid == "" || pPasswd == "") {
    return false;     # empty prepared string
  }

  storedHash = FetchPasswordHash(pAuthcid);
  if (storedHash == NULL || storedHash == "") {
    return false;     # error or unknown authcid
  }

  if (!Compare(storedHash, Hash(pPasswd))) {
    return false;     # incorrect password
  }

  if (authzid == NULL ) {
    authzid = DeriveAuthzid(pAuthcid);
    if (authzid == NULL || authzid == "") {
        return false; # could not derive authzid
    }
  }

  if (!Authorize(pAuthcid, authzid)) {
    return false;     # not authorized
  }

  return true;
}
```

The second parameter of the SASLprep function, when true, indicates that unassigned code points are allowed in the input. When the SASLprep function is called to prepare the password prior to computing the stored hash, the second parameter would be false.

The second parameter provided to the Authorize function is not prepared by this code. The application-level SASL profile should be consulted to determine what, if any, preparation is necessary.

Note that the DeriveAuthzid and Authorize functions (whether implemented as one function or two, whether designed in a manner in which these functions or whether the mechanism implementation can be reused elsewhere) require knowledge and understanding of mechanism and the application-level protocol specification and/or implementation details to implement.

Note that the Authorize function outcome is clearly dependent on details of the local authorization model and policy. Both functions may be dependent on other factors as well.

5. Examples

This section provides examples of PLAIN authentication exchanges. The examples are intended to help the readers understand the above text. The examples are not definitive.

"C:" and "S:" indicate lines sent by the client and server respectively. "<NULL>" represents a single NUL (U+0000) character. The Application Configuration Access Protocol ([ACAP]) is used in the examples.

The first example shows how the PLAIN mechanism might be used for user authentication.

```
S: * ACAP (SASL "CRAM-MD5") (STARTTLS)
C: a001 STARTTLS
S: a001 OK "Begin TLS negotiation now"
<TLS negotiation, further commands are under TLS layer>
S: * ACAP (SASL "CRAM-MD5" "PLAIN")
C: a002 AUTHENTICATE "PLAIN"
S: + ""
C: {21}
C: <NULL>tim<NULL>tanstaaftanstaaf
S: a002 OK "Authenticated"
```

The second example shows how the PLAIN mechanism might be used to attempt to assume the identity of another user. In this example, the server rejects the request. Also, this example makes use of the protocol optional initial response capability to eliminate a round-trip.

```
S: * ACAP (SASL "CRAM-MD5") (STARTTLS)
C: a001 STARTTLS
S: a001 OK "Begin TLS negotiation now"
<TLS negotiation, further commands are under TLS layer>
S: * ACAP (SASL "CRAM-MD5" "PLAIN")
C: a002 AUTHENTICATE "PLAIN" {20+}
C: Ursel<NULL>Kurt<NULL>xipj3plmq
S: a002 NO "Not authorized to requested authorization identity"
```

6. Security Considerations

As the PLAIN mechanism itself provided no integrity nor confidentiality protections, it should not be used without adequate external data security protection, such as TLS services provided by many application layer protocols. By default, implementations SHOULD NOT advertise and SHOULD NOT make use of the PLAIN mechanism unless adequate data security services are in place.

When the PLAIN mechanism is used, the server gains the ability to impersonate the user to all services with the same password regardless of any encryption provided by TLS or other confidentiality protection mechanisms. While many other authentication mechanisms have similar weaknesses, stronger SASL mechanisms address this issue. Clients are encouraged to have an operational mode where all mechanisms which are likely to reveal the user's password to the server are disabled.

General [SASL] security considerations apply to this mechanism.

Unicode, [UTF-8], and [StringPrep] security considerations also apply.

7. IANA Considerations

It is requested that the SASL Mechanism registry [IANA-SASL] entry for the PLAIN mechanism be updated to reflect that this document now provides its technical specification.

```
To: iana@iana.org
Subject: Updated Registration of SASL mechanism PLAIN

SASL mechanism name: PLAIN
Security considerations: See RFC XXXX.
Published specification (optional, recommended): RFC XXXX
Person & email address to contact for further information:
     Kurt Zeilenga <kurt@openldap.org>
     IETF SASL WG <ietf-sasl@imc.org>
Intended usage: COMMON
Author/Change controller: IESG <iesg@ietf.org>
Note: Updates existing entry for PLAIN
```

8. Acknowledgment

This document is a revision of RFC 2595 by Chris Newman. Portions of the grammar defined in Section 2 were borrowed from [UTF-8] by Francois Yergeau.

This document is a product of the IETF Simple Authentication and Security Layer (SASL) Working Group.

9. Normative References

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

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

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

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

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

[Unicode] The Unicode Consortium, "The Unicode Standard, Version 3.2.0" is defined by "The Unicode Standard, Version 3.0" (Reading, MA, Addison-Wesley, 2000. ISBN 0-201-61633-5), as amended by the "Unicode Standard Annex #27: Unicode 3.1" (http://www.unicode.org/reports/tr27/) and by the "Unicode Standard Annex #28: Unicode 3.2" (http://www.unicode.org/reports/tr28/).

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

[TLS] Dierks, T. and, E. Rescorla, "The TLS Protocol Version 1.1", RFC 4346, April 2006.

10. Informative References

[ACAP] Newman, C. and J. Myers, "ACAP -- Application Configuration Access Protocol", RFC 2244, November 1997.

[CRAM-MD5] Nerenberg, L., "The CRAM-MD5 SASL Mechanism", draft-ietf-sasl-crammd5-xx.txt, a work in progress.

[DIGEST-MD5] Leach, P., C. Newman, and A. Melnikov, "Using Digest Authentication as a SASL Mechanism", draft-ietf-sasl-rfc2831bis-xx.txt, a work in progress.

[IANA-SASL] IANA, "SIMPLE AUTHENTICATION AND SECURITY LAYER (SASL) MECHANISMS", <http://www.iana.org/assignments/sasl-mechanisms>.

[SMTP-AUTH] Myers, J., "SMTP Service Extension for Authentication", RFC 2554, March 1999.

11. Editor's Address

Kurt D. Zeilenga
OpenLDAP Foundation

Email: Kurt@OpenLDAP.org

Appendix A. Changes since RFC 2595

This appendix is non-normative.

This document replaces Section 6 of RFC 2595.

The specification details how the server is to compare client-provided character strings with stored character strings.

The ABNF grammar was updated. In particular, the grammar now allows LINE FEED (U+000A) and CARRIAGE RETURN (U+000D) characters in the authzid, authcid, passwd productions. However, whether these control characters may be used depends on the string preparation rules applicable to the production. For passwd and authcid productions, control characters are prohibited. For authzid, one must consult the application-level SASL profile. This change allows PLAIN to carry all possible authorization identity strings allowed in SASL.

Pseudo-code was added.

The example section was expanded to illustrate more features of the PLAIN mechanism.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Full Copyright

Copyright (C) The Internet Society (2006).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.