INTERNET-DRAFT
Intended Category: Experimental
Expires in six months

Kurt D. Zeilenga
Isode Limited
30 May 2009

SASL Yet Another Password Mechanism
<draft-zeilenga-sasl-yap-06.txt>

Status of this Memo

Abstract

This document describes a password authentication mechanism, called

YAP-SHA-256-TLS-UNIQ, for use in protocols which support Simple Authentication and Security Layer (SASL) framework. The mechanism relies on security services provided by a lower layer, such as Transport Layer Security (TLS), to protect the authentication exchange, and subsequent application data exchange, from common attacks. The YAP-SHA-256-TLS-UNIQ mechanism can be viewed as an alternative to other password-based SASL mechanism, such as PLAIN, CRAM-MD5, and DIGEST-MD5.

1. Introduction

There exist multiple password-based mechanisms for use in the Simple Authentication and Security Layer (SASL) [RFC4422] framework. These include the PLAIN [RFC4616], CRAM-MD5 [RFC2195], and DIGEST-MD5 [RFC2831]. None of these mechanisms, themselves, provide integrity and confidential protection over the entirety of the authentication exchange. Only DIGEST-MD5 offers a security layer and, even so, the specification and its implementations suffer from multiple problems. And while these mechanisms may be used in conjunction with lower-level security services, these mechanism do not offer any facility to bind the channels [RFC5056].

This situation has lead to multiple efforts to design "better" SASL password-based mechanism. This document not only specifies yet another password mechanism, YAP-SHA-256-TLS-UNIQ, but defines a family of related password mechanisms, YAP-*.

YAP-* is a family of simple password SASL mechanisms based upon the Keyed-Hash Message Authentication Code (HMAC) [RFC2104] algorithm and unique channel bindings [RFC5056].

The YAP-SHA256-TLS-UNIQ is a YAP mechanism which uses the SHA-256 [FIPS180-2] cryptographic hash function in conjunction with the HMAC algorithm and the tls-unique [CBT-TLS-U] unique channel bindings.

YAP is specified as a family of SASL mechanisms to provide hash agility and channel binding type agility.

YAP mechanisms rely on services provided at a lower level, such as Transport Layer Security (TLS) [RFC5246], to secure the authentication exchange and subsequent application data exchange and, hence, YAP mechanisms do not offer a SASL security layer. YAP mechanisms require the lower-level security layer to be bound in the authentication using unique channel bindings [RFC5056]. YAP relies on client to authenticate the server within this lower-level security layer to avoid information disclosure to rogue servers.

1.1 Experimental

This specification is part of a research and development effort exploring alternatives to current password-based authentication mechanisms.

Implementors of this specification ought to considered implementing the SCRAM [SCRAM] mechanism being developed by the IETF for publication on the Standards Track.

1.2 Terminology

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. The YAP-* Family of Mechanisms

Each mechanism in this family differs by the choice of hash algorithm and the choice of unique channel binding type. Each mechanism has a name of the form YAP-HA-CBT where HA is a string chosen to reflect the hash algorithm used and CBT is a string choosen to reflect the channel binding type. HA and CBT are to be choose so the mechanism name does not exceed 20 characters imposed by the SASL Technical Specification [RFC4422]. While it not required that each mechanism use the same HA string for a particular hash algorithm or the same CBT for a particular channel binding type as those used in previously registered mechanisms, reuse of the encouraged.

To define a new mechanism within the YAP family of mechanisms, the mechanism specification must indicate that it is a YAP mechanism, identify the hash algorithm used, identify the channel binding type used and specify the name the mechanism and cause this name to be registered with IANA in accordance with the SASL Technical Specification. The mechanism specification should detail security considerations specific to hash algorithm and channel binding types selected.

3. The YAP Mechanism

The mechanism involves a single message from the client to the server.

```
message = authzid separator [ authcid ] separator data
separator = %x00
```

where:

- <authzid>, when present, is the authorization identity in the form specified by the application protocol specification, represented in UTF-8 [RFC3629], and

- <authcid> is authentication identity, a simple user name [RFC4013], prepared using the SASLprep [RFC4013] and represented in UTF-8 [RFC3629],

- <data> is a Keyed-Hash Message Authentication Code (HMAC) [RFC2104] produced as described below.

Implementors should note that the data portion of the message may contain a zero-valued octet and hence should parse the message front-to-back.

The HMAC is produced using the mechanism-specific hash algorithm, such as SHA-256, as the cryptographic hash function, H. The secret key, K, is the unique channel binding [RFC5056] for the lower-level security protocol, padded with zero octets to the block size of the hash function. Where the unique channel binding is longer than the block size of the hash function, K is hash of the unique channel binding. The text is the concatenation of the authcid, the authzid, and the hash of the user's password, a simple password [RFC4013], prepared using SASLprep [RFC4013] and represented in UTF-8 [RFC3629]. That is, the <data> is computed as illustrated by the following pseudo code.

```
HMAC(
  Pad( Length(ChannelBinding)>HashBlockSize
    ? H(ChannelBinding) : ChannelBinding, 0, HashBlockSize),
  Concat(authzid, authcid, H(UTF8(SASLprep(password)))))
```

Note, in this pseudo code, the first argument of the HMAC function is the secret key and the second is the text. The cryptographic hash function used in the HMAC is implicitly H. The Pad function pads the first argument to the length specified in the third argument with the octet value provided in the second argument. The variable HashBlockSize is the block size of hash function, H. The Length function returns the length of its argument. The Concat function returns an octet which is the concatenation of its arguments. The UTF8 function returns the UTF-8 encoding of its argument. The SASLprep function prepares it argument according to the SASLprep algorithm. The H function returns the hash of its argument.

The hash of the user's password is a password equivalent. Servers may choose to store this hash instead of the user's password. In either case, the stored value must be adequately protected.

Implementations SHOULD NOT advertise availability of any mechanism in this family unless a lower-level security service providing both data integrity and data confidentiality protection is in place. Client implementations SHOULD NOT utilize any mechanism in this family without first verifying the identity of the server within the lower-level security service. Client implementors should consult the application protocol specification, in conjunction with the specification of the lower-level security service, for details on how to implement this verification.

4. The YAP-SHA-256-TLS-UNIQ Mechanism

The YAP-SHA-256-TLS-UNIQ mechanism is a YAP mechanism which utilizes the SHA-256 [FIPS180-2][RFC4634] hash algorithm and the tls-unique unique channel binding type [CBT-TLS-U]. This type is for use with Transport Layer Security (TLS) [RFC5246].

The mechanism is named "YAP-SHA-256-TLS-UNIQ".

5. YAP-SHA-256-TLS-UNIQ Example

Consider a client authenticating as "kurt" with the password "secret" who is not wishing to act as another user which has established TLS channel which has the tls-unique binding of zHsxigXXUssRg9iVRbw5AX/dgRVlUgBz/RfjI7c4woM= (base64).

The client compute the HMAC over the authzid, authcid, and password as described in section 3 with the binding as the secret key. The client would construct text input the HMAC by concatenating the empty authzid string with the "kurt" authcid string with the hash of the properly prepared password. SASLprep("secret") returns "secret". The hash of this string (when encoded as UTF-8), is K7gNU3sdo+OL0wNhqoVWhr3g6s1xYv72ol/pe/Unols= (base64). The HMAC for this text and key is Ksarn7PFnqCgi4ewSYOfXIyP8ImNcmpoWmtCgA0QqT4= (base64).

The client would construct and send the message which contained first zero octets for the authzid, then a 00 (hex) octet for a separator, followed by 4 octets 6b 75 72 74 (hex) representing the authcid "kurt", followed by 00 (hex) octet for a separator, followed by 32 octets HMAC value. This would produce a message of AGt1cnQAKsarn7PFnqCgi4ewSYOfXIyP8ImNcmpoWmtCgA0QqT4= (base64).

6. Security Considerations

Security is discussed throughout this document.

This family of mechanisms was specifically designed to rely on security services offered at lower-levels to secure mechanism negotiation, the authentication exchange and subsequent data exchanges. To ensure lower-level security services are provided end-to-end, the mechanisms utilize unique channel bindings [RFC5056].

To avoid disclosing the identity information to a rogue server, the client verifies the server's identity using the lower-layer security service before utilizing any mechanism in this family.

Hash agility and channel binding type agility is provided in the family of mechanisms through the specification of additional mechanisms.

To avoid requiring server implementations maintain access to the user's password, a password equivalent is used. The password equivalent is a simple hash of the password.

While it is likely that those choosing to store the password equivalent instead of the password would prefer the equivalent be designed to hinder dictionary attack with precomputed dictionary entries, a simple hash was chosen to avoid adding a server challenge. Use of the authcid as a salt was considered but rejected as it would tie the password equivalent to a particular authcid. It is desirable for the password equivalent to be usable with multiple authcid values (kurt and KURT) representing the same entity. It was also realized that it likely that implementors would (continue to) choose to store the password instead of a mechanism-specific password equivalent. Storing the password avoids significant implementation complexity and facilitates mechanism agility.

YAP-SHA-256-TLS-UNIQ uses the SHA-256 hash algorithm and tls-unique channel binding type. At the time of this writing, there are no known attacks on SHA-256 hash algorithm or tls-unique channel binding type which are applicable to this mechanism.

7. IANA Considerations

It is requested that IANA process the following request(s) upon approval of this document for publication as an RFC.

```
Subject: Registration of SASL YAP family of mechanisms
SASL family name (or prefix for the family): YAP-*
Security considerations: see RFC XXXX
Published specification (recommended): RFC XXXX
```

Person & email address to contact for further information:
Kurt Zeilenga <kurt.zeilenga@isode.com>
Intended usage: COMMON

Subject: Registration of SASL YAP SHA-256 tls_endpoint mechanism
SASL name (or prefix for the family): YAP-SHA-256-TLS-UNIQ
Security considerations: see RFC XXXX
Published specification (recommended): RFC XXXX
Person & email address to contact for further information:
Kurt Zeilenga <kurt.zeilenga@isode.com>
Intended usage: COMMON

7. Acknowledgments

TBD.

8. Author's Address

Kurt D. Zeilenga
Isode Limited

Email: Kurt.Zeilenga@Isode.COM

9. References

[[Note to the RFC Editor: please replace the citation tags used in referencing Internet-Drafts with tags of the form RFCnnnn where possible.]]

9.1. Normative References

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

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

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

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

[RFC4422] Melnikov, A. (Editor), K. Zeilenga (Editor), "Simple

Authentication and Security Layer (SASL)", RFC 4422, June 2006.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[RFC5056] Williams, N., "on the Use of Channel Bidnings to Secure Channels", RFC 5056, November 2007.

[FIPS180-2] National Institute of Standards and Technology, "Secure Hash Algorithm. NIST FIPS 180-2", August 2002.

[CBT-TLS-U] Zhu, Larry, "Registration of TLS unique channel bindings (generic)", http://www.iana.org/assignments/channel-binding-types/tls-unique, June 26, 2008.

9.2. Informative References

[RFC2195] Klensin, J., R. Catoe, and P. Krumviede, "IMAP/POP AUTHorize Extension for Simple Challenge/Response", RFC 2195, September 1997.

[RFC2831] Leach, P. and C. Newman, "Using Digest Authentication as a SASL Mechanism", RFC 2831, May 2000.

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

[RFC4634] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and HMAC-SHA)", RFC 4634, August 2006.

[RFC5246] Dierks, T. and, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[SCRAM] Menon-Sen, A., el. al, "Salted Challenge Response (SCRAM) SASL Mechanism", draft-ietf-sasl-scram-xx.txt, a work in progress.

Full Copyright