Workgroup:
Common Authentication Technology Next
Generation
Internet-Draft:
draft-ietf-kitten-password-storage-04
Published: 18 March 2021
Intended Status: Best Current Practice
Expires: 19 September 2021
Authors: S. Whited

Best practices for password hashing and storage

Abstract

This document outlines best practices for handling user passwords and other authenticator secrets in client-server systems making use of SASL.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on 19 September 2021.

Table of Contents

1. Introduction

Following best practices when hashing and storing passwords for use with SASL impacts a great deal more than just a user's identity. It also affects usability, backwards compatibility, and interoperability by determining what authentication and authorization mechanisms can be used.

1.1. Conventions and Terminology

Various security-related terms are to be understood in the sense defined in [RFC4949]. Some may also be defined in [NISTSP63-3] Appendix A.1 and in [NISTSP132] section 3.1.

Throughout this document the term "password" is used to mean any password, passphrase, PIN, or other memorized secret.

Other common terms used throughout this document include:

Mechanism pinning A security mechanism which allows SASL clients to resist downgrade attacks. Clients that implement mechanism pinning remember the perceived strength of the SASL mechanism used in a previous successful authentication attempt and

thereafter only authenticate using mechanisms of equal or higher perceived strength.

Pepper A secret added to a password hash like a salt. Unlike a salt, peppers are secret and the same pepper may be reused for many hashed passwords. They must not be stored alongside the hashed password.

Salt In this document salt is used as defined in [RFC4949].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. SASL Mechanisms

For clients and servers that support password based authentication using SASL [RFC4422] it is **RECOMMENDED** that the following mechanisms be implemented:

* SCRAM-SHA-256 [RFC7677]

* SCRAM-SHA-256-PLUS [RFC7677]

System entities **SHOULD NOT** invent their own mechanisms that have not been standardized by the IETF or another reputable standards body. Similarly, entities **SHOULD NOT** implement any mechanism with a usage status of "OBSOLETE", or "LIMITED" in the IANA SASL Mechanisms Registry [IANA.sasl.mechanisms] and **MUST NOT** implement any mechanisms with a status of "MUST NOT be used". For example, entities **SHOULD NOT** implement DIGEST-MD5 (deprecated in [RFC6331]).

The SASL mechanisms discussed in this document do not negotiate a security layer. Because of this a strong security layer such as TLS [RFC8446] **MUST** be negotiated before SASL mechanisms can be advertised or negotiated.

3. Client Best Practices

3.1. Mechanism Pinning

Clients often maintain a list of preferred SASL mechanisms, generally ordered by perceived strength to enable strong authentication. To prevent downgrade attacks by a malicious actor that has successfully man in the middled a connection, or compromised a trusted server's configuration, clients **SHOULD** implement "mechanism pinning". That is, after the first successful authentication with a strong mechanism, clients **SHOULD** make a record

of the authentication and thereafter only advertise and use mechanisms of equal or higher perceived strength.

The following mechanisms are ordered by their perceived strength from strongest to weakest with mechanisms of equal strength on the same line. The remainder of this section is merely informative. In particular this example does not imply that mechanisms in this list should or should not be implemented.

1. EXTERNAL

2. SCRAM-SHA-256-PLUS

3. SCRAM-SHA-1-PLUS

4. SCRAM-SHA-256

5. SCRAM-SHA-1

6. PLAIN

The EXTERNAL mechanism defined in [RFC4422] appendix A is placed at the top of the list. However, its perceived strength depends on the underlying authentication protocol. In this example, we assume that TLS [RFC8446] services are being used.

The channel binding ("-PLUS") variants of SCRAM [RFC5802] are listed above their non-channel binding cousins, but may not always be available depending on the type of channel binding data available to the SASL negotiator.

Finally, the PLAIN mechanism sends the username and password in plain text and therefore requires a strong security layer such as TLS for the password to be protected in transit. However, if the server is trusted to know the password PLAIN does allow for the use of a strong key derivation function (KDF) for storing the authentication data at rest and provides for password hash agility.

3.2. Storage

Clients **SHOULD** always store authenticators in a trusted and encrypted keystore such as the system keystore, or an encrypted store created specifically for the clients use. They **SHOULD NOT** store authenticators as plain text.

If clients know that they will only ever authenticate using a mechanism such as SCRAM [RFC5802] where the original password is not needed after the first authentication attempt they **SHOULD** store the SCRAM bits or the hashed and salted password instead of the original password. However, if backwards compatibility with servers that only

support the PLAIN mechanism or other mechanisms that require using the original password is required, clients **MAY** choose to store the original password so long as an appropriate keystore is used.

4. Server Best Practices

4.1. Additional SASL Requirements

Servers **MUST NOT** support any mechanism that would require authenticators to be stored in such a way that they could be recovered in plain text from the stored information. This includes mechanisms that store authenticators using reversable encryption, obsolete hashing mechanisms such as MD5 or hashing mechanisms that are cryptographically secure but designed for speed such as SHA256.

4.2. Storage

Servers **MUST** always store passwords only after they have been salted, peppered (if possible with the given authentication mechanism), and hashed using a strong KDF. A distinct salt **SHOULD** be used for each user, and each SCRAM family supported. Salts **SHOULD** be generated using a cryptographically secure random number generator. The salt **MAY** be stored in the same datastore as the password. A pepper stored in the application configuration, or a secure location other than the datastore containing the salts, **SHOULD** be combined with the password before hashing if possible with the given authentication mechanism. Peppers **SHOULD NOT** be combined with the salt because the salt is not secret and may appear in the final hash output.

The following restrictions MUST be observed when generating salts and peppers, more up to date numbers may be found in [OWASP.CS.passwords].

Parameter	Value
Minimum Salt Length	16 bytes
Minimum Pepper Length	32 bytes

Table 1: Common Parameters

4.3. Authentication and Rotation

When authenticating using PLAIN or similar mechanisms that involve transmitting the original password to the server the password **MUST** be hashed and compared against the salted and hashed password in the database using a constant time comparison.

Each time a password is changed a new random salt **MUST** be created and the iteration count and pepper (if applicable) **MUST** be updated to the latest value required by server policy.

If a pepper is used, consideration should be taken to ensure that it can be easily rotated. For example, multiple peppers could be stored. New passwords and reset passwords would use the newest pepper and a hash of the pepper using a cryptographically secure hash function such as SHA256 could then be stored in the database next to the salt so that future logins can identify which pepper in the list was used. This is just one example, pepper rotation schemes are outside the scope of this document.

5. KDF Recommendations

When properly configured, the following commonly used KDFs create suitable password hash results for server side storage. The recommendations in this section may change depending on the hardware being used and the security level required for the application.

With all KDFs proper tuning is required to ensure that it meets the needs of the specific application or service. For persistent login an iteration count or work factor that adds approximately a quarter of a second to login may be an acceptable tradeoff since logins are relatively rare. By contrast, verification tokens that are generated many times per second may need to use a much lower work factor.

5.1. Argon2

Argon2 [ARGON2ESP] is the 2015 winner of the Password Hashing Competition. Security considerations, test vectors, and parameters for tuning argon2 can be found in [I-D.irtf-cfrg-argon2]. They are copied here for easier reference.

Parameter	Value
Degree of parallelism (p)	1
Minimum memory size (m)	32*1024
Minimum number of iterations (t)	1
Algorithm type (y)	Argon2id (2)
Minimum output length	32

Table 2: Argon Parameters

5.2. Bcrypt

bcrypt [BCRYPT] is a Blowfish-based KDF that is the current OWASP recommendation for password hashing.

Parameter	Value
Recommended Cost	12
Maximum Password Length	50-72 bytes depending on the implementation

Table 3: Bcrypt Parameters

5.3. PBKDF2

PBKDF2 [RFC8018] is used by the SCRAM [RFC5802] family of SASL mechanisms.

Parameter	Value
Minimum iteration count (c)	10,000
Hash	SHA256
Output length (dkLen)	min(hLen, 32) (where hLen is the length of the chosen hash)

Table 4: PBKDF2 Parameters

When PBKDF2 is used with HMAC such as in the SCRAM [RFC5802] family of SASL mechanisms the password is pre-hashed if it is longer than the block size of the hash function (hLen, or 64 bytes for SHA-256). Care should be taken to ensure that the implementation of PBKDF2 does this before the iterations, otherwise long hashes may become significantly more expensive than expected, possibly resulting in a Denial-of-Service (DOS).

5.4. Scrypt

The [SCRYPT] KDF is designed to be memory-hard and sequential memory-hard to prevent against custom hardware based attacks.

Security considerations, test vectors, and further notes on tuning scrypt may be found in [RFC7914].

Parameter	Value
Minimum CPU/Memory cost parameter (N)	32768 (N=2^15)
Blocksize (r)	8
Parallelization parameter (p)	1
Minimum output length (dkLen)	32

Table 5: Scrypt Parameters

6. Password Complexity Requirements

Before any other password complexity requirements are checked, the preparation and enforcement steps of the OpaqueString profile of [RFC8265] **SHOULD** be applied (for more information see the Internationalization Considerations section). Entities **SHOULD** enforce a minimum length of 8 characters for user passwords. If using a mechanism such as PLAIN where the server performs hashing on the original password, a maximum length between 64 and 128 characters **MAY** be imposed to prevent denial of service (DoS) attacks. Entities **SHOULD NOT** apply any other password restrictions.

In addition to these password complexity requirements, servers **SHOULD** maintain a password blocklist and reject attempts by a claimant to use passwords on the blocklist during registration or password reset. The contents of this blocklist are a matter of server policy. Some common recommendations include lists of common passwords that are not otherwise prevented by length requirements, and passwords present in known breaches.

7. Internationalization Considerations

The PRECIS framework (Preparation, Enforcement, and Comparison of Internationalized Strings) defined in [RFC8264] is used to enforce internationalization rules on strings and to prevent common application security issues arrising from allowing the full range of Unicode codepoints in usernames, passwords, and other identifiers. The OpaqueString profile of [RFC8265] is used in this document to ensure that codepoints in passwords are treated carefully and consistently. This ensures that users typing certain characters on different keyboards that may provide different versions of the same character will still be able to log in. For example, some keyboards may output the full-width version of a character while other keyboards output the half-width version of the same character. The Width Mapping rule of the OpaqueString profile addresses this and ensures that comparison succeeds and the claimant is able to be authenticated.

8. Security Considerations

This document contains recommendations that are likely to change over time. It should be reviewed regularly to ensure that it remains accurate and up to date. Many of the recommendations in this document were taken from [OWASP.CS.passwords], [NISTSP63b], and [NISTSP132].

The "-PLUS" variants of SCRAM [RFC5802] support channel binding to their underlying security layer, but lack a mechanism for negotiating what type of channel binding to use. In [RFC5802] the tls-unique [RFC5929] channel binding mechanism is specified as the default, and it is therefore likely to be used in most applications that support channel binding. However, in the absence of the TLS extended master secret fix [RFC7627] and the renegotiation indication TLS extension [RFC5746] the tls-unique and tls-server-endpoint channel binding data can be forged by an attacker that can MITM the connection. Before advertising a channel binding SASL mechanism, entities **MUST** ensure that both the TLS extended master secret fix and the renegotiation indication extension are in place and that the connection has not been renegotiated.

For TLS 1.3 [RFC8446] no channel binding types are currently defined. Channel binding SASL mechanisms **MUST NOT** be advertised or negotiated over a TLS 1.3 channel until such types are defined.

9. IANA Considerations

This document has no actions for IANA.

10. References

10.1. Normative References

[IANA.sasl.mechanisms] IETF, "Simple Authentication and Security Layer (SASL) Mechanisms", November 2015, <https://www.iana.org/assignments/sasl-mechanisms/sasl-mechanisms.xhtml>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/info/rfc4949>.

[RFC5746] Rescorla, E., Ray, M., Dispensa, S., and N. Oskov, "Transport Layer Security (TLS) Renegotiation Indication Extension", RFC 5746, DOI 10.17487/RFC5746, February 2010, <https://www.rfc-editor.org/info/rfc5746>.

[RFC5929] Altman, J., Williams, N., and L. Zhu, "Channel Bindings for TLS", RFC 5929, DOI 10.17487/RFC5929, July 2010, <https://www.rfc-editor.org/info/rfc5929>.

[RFC7627] Bhargavan, K., Ed., Delignat-Lavaud, A., Pironti, A., Langley, A., and M. Ray, "Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension", RFC 7627, DOI 10.17487/RFC7627, September 2015, <https://www.rfc-editor.org/info/rfc7627>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

10.2. Informative References

[ARGON2ESP] Biryukov, A., Dinu, D., and D. Khovratovich, "Argon2: New Generation of Memory-Hard Functions for Password Hashing and Other Applications", Euro SnP 2016, March

2016, <https://www.cryptolux.org/images/d/d0/Argon2ESP.pdf>.

[BCRYPT] Provos, N. and D. Mazières, "A Future-Adaptable Password Scheme", USENIX 1999 https://www.usenix.org/legacy/event/usenix99/provos/provos.pdf, June 1999.

[I-D.irtf-cfrg-argon2] Biryukov, A., Dinu, D., Khovratovich, D., and S. Josefsson, "The memory-hard Argon2 password hash and proof-of-work function", Work in Progress, Internet-Draft, draft-irtf-cfrg-argon2-12, 8 September 2020, <https://tools.ietf.org/html/draft-irtf-cfrg-argon2-12>.

[NISTSP132] Turan, M., Barker, E., Burr, W., and L. Chen, "Recommendation for Password-Based Key Derivation Part 1: Storage Applications", NIST Special Publication SP 800-132, DOI 10.6028/NIST.SP.800-132, December 2010, <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf>.

[NISTSP63-3] Grassi, P., Garcia, M., and J. Fenton, "Digital Identity Guidelines", NIST Special Publication SP 800-63-3, DOI 10.6028/NIST.SP.800-63-3, June 2017, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-3.pdf>.

[NISTSP63b] Grassi, P., Fenton, J., Newton, E., Perlner, R., Regenscheid, A., Burr, W., Richer, J., Lefkovitz, N., Danker, J., Choong, Y., Greene, K., and M. Theofanos, "Digital Identity Guidelines: Authentication and Lifecycle Management", NIST Special Publication SP 800-63b, DOI 10.6028/NIST.SP.800-63b, June 2017, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b.pdf>.

[OWASP.CS.passwords] Manico, J., Saad, E., Maćkowski, J., and R. Bailey, "Password Storage", OWASP Cheat Sheet Password Storage, April 2020, <https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html>.

[RFC4422] Melnikov, A., Ed. and K. Zeilenga, Ed., "Simple Authentication and Security Layer (SASL)", RFC 4422, DOI 10.17487/RFC4422, June 2006, <https://www.rfc-editor.org/info/rfc4422>.

[RFC5802] Newman, C., Menon-Sen, A., Melnikov, A., and N. Williams, "Salted Challenge Response Authentication Mechanism (SCRAM) SASL and GSS-API Mechanisms", RFC 5802, DOI

10.17487/RFC5802, July 2010, <https://www.rfc-editor.org/info/rfc5802>.

[RFC6331] Melnikov, A., "Moving DIGEST-MD5 to Historic", RFC 6331, DOI 10.17487/RFC6331, July 2011, <https://www.rfc-editor.org/info/rfc6331>.

[RFC7677] Hansen, T., "SCRAM-SHA-256 and SCRAM-SHA-256-PLUS Simple Authentication and Security Layer (SASL) Mechanisms", RFC 7677, DOI 10.17487/RFC7677, November 2015, <https://www.rfc-editor.org/info/rfc7677>.

[RFC7914] Percival, C. and S. Josefsson, "The scrypt Password-Based Key Derivation Function", RFC 7914, DOI 10.17487/RFC7914, August 2016, <https://www.rfc-editor.org/info/rfc7914>.

[RFC8018] Moriarty, K., Ed., Kaliski, B., and A. Rusch, "PKCS #5: Password-Based Cryptography Specification Version 2.1", RFC 8018, DOI 10.17487/RFC8018, January 2017, <https://www.rfc-editor.org/info/rfc8018>.

[RFC8264] Saint-Andre, P. and M. Blanchet, "PRECIS Framework: Preparation, Enforcement, and Comparison of Internationalized Strings in Application Protocols", RFC 8264, DOI 10.17487/RFC8264, October 2017, <https://www.rfc-editor.org/info/rfc8264>.

[RFC8265] Saint-Andre, P. and A. Melnikov, "Preparation, Enforcement, and Comparison of Internationalized Strings Representing Usernames and Passwords", RFC 8265, DOI 10.17487/RFC8265, October 2017, <https://www.rfc-editor.org/info/rfc8265>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[SCRYPT] Percival, C., "Stronger key derivation via sequential memory-hard functions", BSDCan'09 http://www.tarsnap.com/scrypt/scrypt.pdf, May 2009.

Appendix A. Acknowledgments

The author would like to thank the civil servants at the National Institute of Standards and Technology for their work on the Special Publications series. U.S. executive agencies are an undervalued national treasure, and they deserve our thanks.

Thanks also to Cameron Paul and Thomas Copeland for their reviews and suggestions.

Author's Address

Sam Whited
Atlanta, GA
United States of America

Email: sam@samwhited.com
URI: https://blog.samwhited.com/