XEP-0438: Best practices for password hashing and storage

Sam Whited
mailto:sam@samwhited.com
xmpp:sam@samwhited.com
https://blog.samwhited.com/

2020-10-30
Version 0.2.0

Status	Type	Short Name
Experimental	Informational	passwords

This document outlines best practices for handling user passwords on the public Jabber network for both clients and servers.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

Following best practices when hashing and storing passwords and other authenticator secrets impacts a great deal more than just a users identity. It also effects usability, and backwards compatibility by determining what authentication and authorization mechanisms can be used. Unfortunately, aside from mandating the use of SCRAM-SHA-1 in RFC 6120 [1], and recommending at least 4096 rounds of PBKDF2 in RFC 5802 [2] (a number which is now woefully inadequate), no general recommendations for best practices in password storage, transmission, or key derivation function tuning exist in the XMPP ecosystem.
Many of the recommendations in this document were taken from Digital Identity Guidelines: Authentication and Lifecycle Management [3] and Recommendation for Password-Based Key Derivation, Part 1: Storage Applications [4].

2 Requirements

This document makes specific recommendations for best practices on the public Jabber network for both clients and servers. It does not attempt to address private networks or proprietary services which may have different requirements, use cases, and security models. These recommendations include the hashing and storage of memorized secrets and other authenticators, authentication, and compatibility between clients and servers with respect to authentication.
To keep the length of this document manageable, we assume basic familiarity with password storage and handling, common terms, and cryptographic operations. For an overview of basic password security see the OWASP Password Storage Cheat Sheet [5] maintained by the Open Web Application Security Project (OWASP) [6].

3 Glossary

Various security-related terms are to be understood in the sense defined in RFC 4949 [7] Some may also be defined in defined in Digital Identity Guidelines [8] Appendix A.1 and in

[1] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.
[2] RFC 5802: Salted Challenge Response Authentication Mechanism (SCRAM) SASL and GSS-API Mechanisms <http://tools.ietf.org/html/rfc5802>.
[3] Digital Identity Guidelines: Authentication and Lifecycle Management, NIST Special Publication 800-63B <https://doi.org/10.6028/NIST.SP.800-63b>.
[4] Recommendation for Password-Based Key Derivation, Part 1: Storage Applications, NIST Special Publication 800-132 <https://doi.org/10.6028/NIST.SP.800-132>.
[5] OWASP Cheat Sheet Series for password storage <https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html>.
[6] The Open Web Application Security Project (OWASP, or OWASP Foundation) is a nonprofit foundation that works to improve the security of software. For further information, see <https://owasp.org/>.
[7] RFC 4949: Internet Security Glossary, Version 2 <http://tools.ietf.org/html/rfc4949>.
[8] Digital Identity Guidelines, NIST Special Publication 800-63-3 <https://doi.org/10.6028/NIST.SP.800-63-3>.

Recommendation for Password-Based Key Derivation, Part 1: Storage Applications [9] §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 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 not unique. They must not be stored alongside the hashed password.

Salt In this document salt is used as defined in RFC 4949 RFC 4949: Internet Security Glossary, Version 2 <http://tools.ietf.org/html/rfc4949>..

4 SASL Mechanisms

Clients and servers must already implement the SASL mechanisms listed in RFC 6120 §13.8.1 For Authentication Only. These mechanisms are:

- SCRAM-SHA-1
- SCRAM-SHA-1-PLUS

In addition, clients and servers SHOULD support the following SCRAM variants defined in RFC 7677 [10]:

- SCRAM-SHA-256
- SCRAM-SHA-256-PLUS

Clients SHOULD NOT invent their own mechanisms that have not been standardized by the IETF, the XSF, or another reputable standards body.
Clients MUST NOT implement any mechanism with a usage status of ”OBSOLETE”, ”MUST NOT be used”, or ”LIMITED” in the IANA SASL Mechanisms Registry [11]. Similarly, any mechanism that depends on a hash function listed as ”MUST NOT” in Cryptographic Hash Function Recommendations for XMPP (XEP-0414) [12] MUST NOT be used. This means that the following

[9] Recommendation for Password-Based Key Derivation, Part 1: Storage Applications, NIST Special Publication 800-132 <https://doi.org/10.6028/NIST.SP.800-132>.
[10] RFC 7677: SCRAM-SHA-256 and SCRAM-SHA-256-PLUS Simple Authentication and Security Layer (SASL) Mechanisms <http://tools.ietf.org/html/rfc7677>.
[11] IANA registry of mechanisms used in the Simple Authentication and Security Layer protocol <http://www.iana.org/assignments/sasl-mechanisms>.
[12] XEP-0414: Cryptographic Hash Function Recommendations for XMPP <https://xmpp.org/extensions/xep-0414.html>.

mechanisms which were commonly used with XMPP in the past MUST NOT be supported:

- CRAM-MD5 (RFC 2195 [13])
- DIGEST-MD5 (RFC 6331 [14])

5 Client Best Practices

5.1 Mechanism Pinning

Clients maintain a list of preferred SASL mechanisms, generally ordered by perceived strength to enable strong authentication (RFC 6120 [15] §6.3.3 Mechanism Preferences). 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.
For reference, the following mechanisms are ordered by their perceived strength from strongest to weakest with mechanisms of equal strength on the same line. This list is a non-normative example and does not indicate that these mechanisms should or should not be supported:

1. EXTERNAL
2. SCRAM-SHA-1-PLUS, SCRAM-SHA-256-PLUS
3. SCRAM-SHA-1, SCRAM-SHA-256
4. PLAIN
5. DIGEST-MD5, CRAM-MD5

The EXTERNAL mechanism defined in RFC 4422 [16] 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 (RFC 8446 [17]) services are being used.
The channel binding ("-PLUS") variants of SCRAM (RFC 5802 [18]) are listed above their non-channel binding cousins, but may not always be available depending on the type of channel

[13] RFC 2195: IMAP/POP AUTHorize Extension for Simple Challenge/Response <http://tools.ietf.org/html/rfc2195>.
[14] RFC 6331: Moving DIGEST-MD5 to Historic <http://tools.ietf.org/html/rfc6331>.
[15] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.
[16] RFC 4422: Simple Authentication and Security Layer (SASL) <http://tools.ietf.org/html/rfc4422>.
[17] RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3 <http://tools.ietf.org/html/rfc8446>.
[18] RFC 5802: Salted Challenge Response Authentication Mechanism (SCRAM) SASL and GSS-API Mechanisms <http://tools.ietf.org/html/rfc5802>.

binding data available to the SASL negotiator.
The PLAIN mechanism sends the username and password in plain text, but does allow for the use of a strong key derivation function (KDF) for the stored version of the password on the server.
Finally, the DIGEST-MD5 and CRAM-MD5 mechanisms are listed last because they use weak hashes and ciphers and prevent the server from storing passwords using a KDF. For a list of problems with DIGEST- MD5 see RFC 6331 [19].

5.2 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 where the original password is not needed (for example if the mechanism has been pinned) 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.

6 Server Best Practices

6.1 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, and hashes that are unsuitable for use with authenticators such as SHA256.

6.2 Storage

Servers MUST always store passwords only after they have been salted and hashed using a strong KDF. If multiple hashes are supported for use with SCRAM, for example SCRAM-SHA-1 and SCRAM-SHA-256, separate salted and hashed passwords SHOULD be calculated and stored for each mechanism so that users can log in with multiple clients that support only some of the mechanisms.
A distinct salt SHOULD be used for each user, and each SCRAM family supported. Salts MUST be generated using a cryptographically secure random number generator. The salt MAY be stored in the same datastore as the password. If it is stored alongside the password, it SHOULD

[19] RFC 6331: Moving DIGEST-MD5 to Historic <http://tools.ietf.org/html/rfc6331>.

be combined with a pepper stored in the application configuration, an environment variable, or some other location other than the datastore containing the salts.
The following minimum restrictions MUST be observed when generating salts and peppers. More up to date numbers may be found in OWASP Password Storage Cheat Sheet [20]

Minimum Salt Length 16 bytes

Minimum Pepper Length 32 bytes

6.3 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 or reset, a new random salt should be created and the iteration count and pepper (if applicable) should 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 with new passwords and reset passwords using the latest pepper. 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.

7 PBKDF2 Parameters

Because the PBKDF2 key derivation function (RFC 8018 [21]) is used by SCRAM-SHA-1 which is mandated for use in XMPP, this document recommends it for password storage. Servers SHOULD use the following parameters when applying PBKDF2:

Minimum iteration count (c) 10,000 (100,000 for higher security environments)

Hash SHA256

Output length (dkLen) hLen (length of the chosen hash)

The minimum iteration count may be tuned to the specific system on which password hashing is taking place.

[20] OWASP Cheat Sheet Series for password storage <https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html>.
[21] RFC 8018: PKCS #5: Password-Based Cryptography Specification Version 2.1 <http://tools.ietf.org/html/rfc8018>.

8 Password Complexity Requirements

Before any other password complexity requirements are checked, the preparation and enforcement steps of the OpaqueString profile of RFC 8265 [22] 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.

9 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 Password Storage Cheat Sheet [23], Digital Identity Guidelines: Authentication and Lifecycle Management [24], and Recommendation for Password-Based Key Derivation, Part 1: Storage Applications [25].
The SCRAM suite of SASL mechanisms are recommended in this document, however, there is currently no way to force a password reset. This reduces upgrade agility if a weakness is discovered in SCRAM and means that new, untested, SCRAM-based or SCRAM-like mechanisms should be added with caution.
The "-PLUS" variants of SCRAM support channel binding to their underlying security layer, but lack a mechanism for negotiating what type of channel binding to use. In RFC 5802 [26] the tls-unique (RFC 5929 [27]) 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 (RFC 7627 [28]) and the renegotiation indication

[22] RFC 8265: Preparation, Enforcement, and Comparison of Internationalized Strings Representing Usernames and Passwords <http://tools.ietf.org/html/rfc8265>.
[23] OWASP Cheat Sheet Series for password storage <https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html>.
[24] Digital Identity Guidelines: Authentication and Lifecycle Management, NIST Special Publication 800-63B <https://doi.org/10.6028/NIST.SP.800-63b>.
[25] Recommendation for Password-Based Key Derivation, Part 1: Storage Applications, NIST Special Publication 800-132 <https://doi.org/10.6028/NIST.SP.800-132>.
[26] RFC 5802: Salted Challenge Response Authentication Mechanism (SCRAM) SASL and GSS-API Mechanisms <http://tools.ietf.org/html/rfc5802>.
[27] RFC 5929: Channel Bindings for TLS <http://tools.ietf.org/html/rfc5929>.
[28] RFC 7627: Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension <http://tools.ietf.org/html/rfc7627>.

TLS extension (RFC 5746 [29]) 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.
This document mentions many hash functions that are already in use in the XMPP ecosystem, or that have been used in the past. It does not make recommendations for what functions should or should not be used in new applications. For recommendations about the use of hash functions and their security implications, see Cryptographic Hash Function Recommendations for XMPP (XEP-0414) [30]
For TLS 1.3 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.

10 Internationalization Considerations

The PRECIS framework (Preparation, Enforcement, and Comparison of Internationalized Strings) defined in RFC 8264 [31] 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 RFC 8265 [32] 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.

11 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [33].

[29] RFC 5746: Transport Layer Security (TLS) Renegotiation Indication Extension <http://tools.ietf.org/html/rfc5746>.
[30] XEP-0414: Cryptographic Hash Function Recommendations for XMPP <https://xmpp.org/extensions/xep-0414.html>.
[31] RFC 8264: PRECIS Framework: Preparation, Enforcement, and Comparison of Internationalized Strings in Application Protocols <http://tools.ietf.org/html/rfc8264>.
[32] RFC 8265: Preparation, Enforcement, and Comparison of Internationalized Strings Representing Usernames and Passwords <http://tools.ietf.org/html/rfc8265>.
[33] The Internet Assigned Numbers Authority (IANA) is the central coordinator for the assignment of unique parameter values for Internet protocols, such as port numbers and URI schemes. For further information, see

12 XMPP Registrar Considerations

No namespaces or parameters need to be registered with the XMPP Registrar [34] as a result of this document.

<http://www.iana.org/>.

[34] The XMPP Registrar maintains a list of reserved protocol namespaces as well as registries of parameters used in the context of XMPP extension protocols approved by the XMPP Standards Foundation. For further information, see <https://xmpp.org/registrar/>.