XEP-0515: TLS Channel-Binding Downgrade Protection

Thilo Molitor
mailto:thilo+xmpp@eightysoft.de
xmpp:thilo.molitor@juforum.de

2026-06-30
Version 0.1.0

Status	Type	Short Name
Experimental	Standards Track	TDP

This specification provides a way to secure the SASL and SASL2 SCRAM handshakes against channel-binding downgrades through TLS version downgrades.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

SASL SCRAM Downgrade Protection (XEP-0474) [1] defines a way to detect and prevent channel-binding type and SASL method downgrades. While this works well, an attacker could still leverage one specific attack vector to downgrade the channel-binding type to tls-server-end-point: a MITM-attacker could use different TLS versions with a different set of supported channel-binding types on both arms of their connection. This will downgrade the channel-binding type negotiated to the lowest denominator of both lists, which is at least tls-server-end-point per requirement of SASL Channel-Binding Type Capability (XEP-0440) [2].
If the server violates point 1 of the Business Rules in SASL Channel-Binding Type Capability (XEP-0440) [3] and the client simultaneously violates point 6 in said Business Rules, the attacker will even be able to downgrade the connection no channel-binding at all!
While pinning of channel-binding types can prevent those downgrade attacks, pinning comes with all the downsides explained in SASL SCRAM Downgrade Protection (XEP-0474) [4] and won't secure the very first connection either.

1.1 Examples

While this attack doesn't depend on specific TLS versions, but can be executed with all current or future TLS versions for which the list of defined/implemented channel-binding types differes, here are two examples for TLS 1.2 and TLS 1.3.
If the MITM-attacker terminates the TLS connection to the client with TLS 1.3 and the TLS connection to the server with TLs 1.2, the server will advertise tls-uniqe channel-binding (alongside tls-server-end-point), but not tls-exporter. The client won't pick tls-unique, because it isn't defined for TLS 1.3 and fall back to the weaker tls-server-end-point.
If the MITM-attacker chooses to terminate the TLS connection to the client with TLS 1.2 and to the server with TLS 1.3, the server will advertise tls-exporter channel-binding (alongside tls-server-end-point), but not tls-unique. Even though tls-exporter can be securely used if the extended-master-secret extension TLS extension is used, most clients won't pick tls-exporter on TLS 1.2 connections and thus fall back to the weaker tls-server-end-point.

2 Glossary

This specification uses some abbreviations:

- MITM: man-in-the-middle

[1] XEP-0474: SASL SCRAM Downgrade Protection <https://xmpp.org/extensions/xep-0474.html>.
[2] XEP-0440: SASL Channel-Binding Type Capability <https://xmpp.org/extensions/xep-0440.html>.
[3] XEP-0440: SASL Channel-Binding Type Capability <https://xmpp.org/extensions/xep-0440.html>.
[4] XEP-0474: SASL SCRAM Downgrade Protection <https://xmpp.org/extensions/xep-0474.html>.

- SASL1: the XMPP SASL profile specified in RFC 6120 [5]
- SASL2: the XMPP SASL profile specified in Extensible SASL Profile (XEP-0388) [6]
- TDP: TLS downgrade protection, this specification

3 Requirements

This protocol was designed with the following requirements in mind:

- Allow detection of TLS version and thus channel-binding downgrades even if no channel-binding ends-up being used.
- Support all currently defined and future SCRAM mechanisms (RFC 5802 [7] and RFC 7677 [8]).
- Secure the very first connection.
- Be not less secure than pinning when using the SCRAM family of mechanisms (or some similar challenge-response based authentication mechanism).

Note that this specification intentionally leaves out support for SASL PLAIN. If server and client support PLAIN, no protection against SASL method or channel-binding downgrades is possible and the security relies solely on the underlying TLS channel. See the Business Rules section of SASL SCRAM Downgrade Protection (XEP-0474) [9] for some advice on how to handle PLAIN.

4 Protocol

Sections 5.1 and 7 of RFC 5802 [10] allow for arbitrary optional attributes inside SCRAM messages. This specification uses those optional attributes to implement a downgrade protection.

[5] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.
[6] XEP-0388: Extensible SASL Profile <https://xmpp.org/extensions/xep-0388.html>.
[7] RFC 5802: Salted Challenge Response Authentication Mechanism (SCRAM) SASL and GSS-API Mechanisms <http://tools.ietf.org/html/rfc5802>.
[8] RFC 7677: SCRAM-SHA-256 and SCRAM-SHA-256-PLUS Simple Authentication and Security Layer (SASL) Mechanisms <http://tools.ietf.org/html/rfc7677>.
[9] XEP-0474: SASL SCRAM Downgrade Protection <https://xmpp.org/extensions/xep-0474.html>.
[10] RFC 5802: Salted Challenge Response Authentication Mechanism (SCRAM) SASL and GSS-API Mechanisms <http://tools.ietf.org/html/rfc5802>.

4.1 Server Sends TLS Version As Hex

The server encodes the TLS version number used by the connection as defined in RFC 8446 [11], RFC 5246 [12] and their predecessors as four lowercased hexadecimal numbers. For convenience the currently defined version numbers follow:

1. TLS 1.3 has the version number 0x0304, thus is encoded as '0304'
2. TLS 1.2 has the version number 0x0303, thus is encoded as '0303'
3. TLS 1.1 has the version number 0x0302, thus is encoded as '0302'
4. TLS 1.0 has the version number 0x0301, thus is encoded as '0301'

The server then adds the optional attribute "t" with the value of the four hexadecimal characters described above to its server-first-message.

4.2 Client Verifies The TLS Version Number

Upon receiving the server-first-message the client calculates the version number of its own TLS connection and encodes it as four lowercased hex-encoded characters as described in Server Sends TLS Version As Hex.
The client then extracts the TLS version number presented by the server in the optional attribute "t" and compares it to its own version number. If the hex-encoded version numbers match, the TLS version used by the server and client have not been altered by an active MITM. If the version numbers do not match, the client MUST fail the authentication. It MAY additionally show a user-facing warning message about an active MITM. If the version numbers match, an attacker could still have manipulated them. If so, the server will always fail the authentication according to RFC 5802 [13] because the client-proof will not be based upon the correct TDP value.

4.3 Full Example

This sections contains an example based on the ones provided in Extensible SASL Profile (XEP-0388) [14].

Listing 1: Full SCRAM-SHA-1-PLUS authentication flow using the optional attribute defined in this spec

[11] RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3 <http://tools.ietf.org/html/rfc8446>.
[12] RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2 <http://tools.ietf.org/html/rfc5246>.
[13] RFC 5802: Salted Challenge Response Authentication Mechanism (SCRAM) SASL and GSS-API Mechanisms <http://tools.ietf.org/html/rfc5802>.
[14] XEP-0388: Extensible SASL Profile <https://xmpp.org/extensions/xep-0388.html>.

```
<!-{}-
  Client sending stream header
-{}->
<stream:stream
  from='user@example.org'
  to='example.org'
  version='1.0'
  xml:lang='en'
  xmlns='jabber:client'
  xmlns:stream='http://etherx.jabber.org/streams'>

<!-{}-
  Server responding with stream header and features
-{}->
<stream:stream
  from='example.org'
  id='++TR84Sm6A3hnt3Q065SnAbbk3Y='
  to='user@example.org'
  version='1.0'
  xml:lang='en'
  xmlns='jabber:client'
  xmlns:stream='http://etherx.jabber.org/streams'>
<stream:features>
  <authentication xmlns='urn:xmpp:sasl:2'>
    <mechanism>SCRAM-SHA-1</mechanism>
    <mechanism>SCRAM-SHA-1-PLUS</mechanism>
    <inline xmlns='urn:xmpp:sasl:2'>
      <!-{}- Server indicates that XEP-0198 can be negotiated "inline"
          -{}->
      <enable xmlns='urn:xmpp:sm:3'/>
      <!-{}- Server indicates support for XEP-0386 Bind 2 -{}->
      <bind xmlns='urn:xmpp:bind2:1'/>
    </inline>
  </authentication>
  <!-{}- Channel-binding information provided by XEP-0440 -{}->
  <sasl-channel-binding xmlns='urn:xmpp:sasl-cb:0'>
    <channel-binding type='tls-server-end-point'/>
    <channel-binding type='tls-exporter'/>
  </sasl-channel-binding>
</stream:features>

<!-{}-
  Client initiates authentication using SCRAM-SHA-1-PLUS and channel-
     binding type "tls-exporter"
-{}->
<authenticate xmlns='urn:xmpp:sasl:2' mechanism='SCRAM-SHA-1-PLUS'>
  <!-{}- Base64 of: 'p=tls-exporter,,n=user,r=12C4CD5C-E38E-4A98-8F6D
     -15C38F51CCC6' -{}->
  <initial-response>
```

```
      cD10bHMtZXhwb3J0ZXIsLG49dXNlcixyPTEyQzRDRDVDLUUzOEUtNEE5OC04RjZELTE1QzM4RjUxQ0ND
      ==</initial-response>
  <user-agent id='d4565fa7-4d72-4749-b3d3-740edbf87770'>
    <software>AwesomeXMPP</software>
    <device>Kiva's Phone</device>
  </user-agent>
</authenticate>

<!-{}-
  SCRAM-SHA-1-PLUS challenge issued by the server as defined in RFC
    5802 including the hex-encoded TLS version and the hash defined in
    XEP-0474..
  Attribute "t" contains 0304 (TLS 1.3), attribute "h" contains the
    XEP-0474 hash.
  Base64 of: 'r=12C4CD5C-E38E-4A98-8F6D-15C38F51CCC6a09117a6-ac50-4f2f
    -93f1-93799c2bddf6,s=QSXCR+Q6sek8bf92,i=4096,h=G6k/
    rBLDqgOhRRaCuuatSDFkJ08=,t=0304'
-{}->
<challenge xmlns='urn:xmpp:sasl:2'>
  
    cj0xMkM0Q0Q1Qy1FMzhFLTRBOTgtOEY2RC0xNUMzOEY1MUNDQzZhMDkxMTdhNi1hYzUwLTRmMmYtOTNmMS

</challenge>

<!-{}-
  The client responds with the base64 encoded SCRAM-SHA-1-PLUS client-
    final-message (password: 'pencil')
  The c-attribute contains the GS2-header and channel-binding data
    blob as defined in RFC 5802.
  Base64 of: 'c=cD10bHMtZXhwb3J0ZXIsLFRISVMgSVMgRkFLRSBDQiBEQVRB,r=12
    C4CD5C-E38E-4A98-8F6D-15C38F51CCC6a09117a6-ac50-4f2f-93f1-93799
    c2bddf6,p=KHUfN8dSy1K95crT4D5y1ItLJfs='
-{}->
<response xmlns='urn:xmpp:sasl:2'>
  
    Yz1jRDEwYkhNdFpYaHdiM0owWlhJc0xGUklTVk1nU1ZNZ1JrRkxSU0JEUWlCRVFWUkIscj0xMkM0Q0Q1Qy

</response>

<!-{}-
  The server accepted this authentication, no tampering with the
    advertised SASL mechanisms or channel-bindings was detected.
-{}->
<success xmlns='urn:xmpp:sasl:2'>
  <!-{}- Base64 of: 'v=3w34ZIMVRkx2f2Ozb3/ecRPVdv4=' -{}->
  <additional-data>
    dj0zdzM0WklNVlJreDJmMk96YjMvZWNSUFZkdjQ9
  </additional-data>
  <authorization-identifier>user@example.org</authorization-identifier
```

```
    >
</success>
```

5 Business Rules

To implement this protocol, clients and servers MUST also implement SASL SCRAM Downgrade Protection (XEP-0474) [15].
The rules outlined in the Business Rules section of SASL SCRAM Downgrade Protection (XEP-0474) [16] are important and MUST be followed when implementing this specification.

6 Security Considerations

Using SCRAM attributes makes them part of the HMAC signatures used in the SCRAM protocol flow, efficiently protecting them against any MITM attacker not knowing the password used.

7 IETF Interaction

This protocol shall be superseded by any IETF RFC providing some or all of the functionality provided by this specification. If such a specification exists implementations SHOULD NOT implement this XEP and SHOULD implement the superseding RFC instead.

8 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [17].

9 Acknowledgements

Thanks to Holger Weiß and Paweł Chmielowski for their feedback.

[15] XEP-0474: SASL SCRAM Downgrade Protection <https://xmpp.org/extensions/xep-0474.html>.
[16] XEP-0474: SASL SCRAM Downgrade Protection <https://xmpp.org/extensions/xep-0474.html>.
[17] 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 <http://www.iana.org/>.

10 XMPP Registrar Considerations

This specification does not need any interaction with the XMPP Registrar [18].

11 XML Schema

This specification does not specify any new XML elements.

[18] 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/>.