XEP-0400: Multi-Factor Authentication with TOTP

Dave Cridland
mailto:dave@hellopando.com
xmpp:dwd@dave.cridland.net

2018-01-25
Version 0.1.0

Status	Type	Short Name
Deferred	Standards Track	mfa

This specification defines support for multi-factor authentication in terms of SASL2 Tasks based around the Time-based One Time Password mechanism.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

This XMPP Extension Protocol has been contributed in full conformance with the XSF’s Intellectual Property Rights Policy (a copy of which can be found at <https://xmpp.org/about/xsf/ipr-policy> or obtained by writing to XMPP Standards Foundation, P.O. Box 787, Parker, CO 80134 USA).

Contents

1 Introduction

It is generally agreed that the security of passwords can be improved when combining with another factor, such as possession of a hardware token or control of another account, etc. This specification provides a suite of SASL2 tasks (see Extensible SASL Profile (XEP-0388) [1]) and supporting protocol to allow users or administrators to perform such secondary authentication.
This specification currently only discusses use of TOTP with SASL2; it should be noted that if client support is needed, it is far superior to simply support SASL2.
Therefore this specification takes the view that support for entry of TOTP codes where the client has no support should be of an ad-hoc nature, or potentially unsupported by the server.

2 Outline of use

We start by describing the user's device capable of generating TOTP Codes as the TOTP Device. In order to support TOTP, both the TOTP Device and the server are required to have a shared TOTP Secret which can be used to generate the codes according to RFC 6238 [2]. The process by which a TOTP Secret is generated by the server and passed to the TOTP Device is known as enrollment.
In XMPP, enrollment can be initiated either by the server (due to an administrative fiat that the account requires TOTP) or by the user. If it is initiated by the user, the flow is a simple <iq/> based protocol; on the other hand enforced enrollment by the server is performed as a SASL2 Task.
Once enrolled, codes are sent by the client during a second SASL2 Task. Servers might offer this as the only Multifactor option, or as one of many.
Typically, servers supporting MFA also support CLIENT-KEY and/or CLIENT-KEY-PLUS, and will suppress MFA when these are used. This vastly improves user experience for many cases.

3 Interoperability Notes

3.1 Use with naïve clients

Clients which do not have support for TOTP will no longer work on accounts which have been enrolled. This is problematic, and a number of options have been considered, such as sending a plain <message/> stanza during client connection.
However, use of plain messages from the server has been known to train users into bad behaviour and is easily spoofable. Therefore this specification leaves how to support legacy clients open, and proposes that users do not enroll TOTP until all their clients support it.

[1] XEP-0388: Extensible SASL Profile <https://xmpp.org/extensions/xep-0388.html>.
[2] RFC 6238: TOTP: Time-Based One-Time Password Algorithm <http://tools.ietf.org/html/rfc6238>.

3.2 TOTP Parameters

Although TOTP is hash-agile and supports a range of parameters, in practise deployment has been geared heavily toward a single implementation, and therefore practical constraints on the algorithm defined in RFC 6238 [3] are significant.
In particular, the hash algorithm MUST be SHA-1, and the period MUST be 30 seconds. 6 SHALL be the number of digits, and the number of digits SHALL be 6.
A single implementation restricting the use of modern hash algorithms is, of course, bad, but lack of interoperability would be similarly bad.

4 TOTP URIs

A commonly implemented technique for passing TOTP Secrets is to encode them as a URI within which the various parameters, including the TOTP secret, are specified. Unfortunately this URI scheme appears to only be specified on a Wiki page.
However, this URI scheme is so widely supported that interoperability demands that it is used, so this document therefore specifies a cut-down variant of the URI which is to be used within XMPP. Treatment of this URI as anything but an especially formatted string is not within the scope of this document.
A TOTP URI is specified with the following ABNF:

```
totp-uri = "otpauth://totp/" label "?secret=" secret "&issuer=" issuer
label = issuer (":" / "%3A") jid
jid = 1*CHAR ; URI-encoded jid
secret = 40 * HEXCHAR ; Base32 (hex) encoded secret with no padding.
issuer = 1*CHAR ; Issuer name.
```

Yes, issuer is in there twice. No, I don't either.
TOTP URIs are normally presented to the user as a QR Code

5 TOTP Support Operations

5.1 Voluntary Account Enrollment

In order to voluntarily enroll, a client sends an <iq/> of type set containing an empty element <setup/>, qualified by the namespace urn:xmpp:mfa:0.

```
<iq type='set' id='123456'>
  <setup xmlns='urn:xmpp:mfa:0'/>
</iq>
```

[3] RFC 6238: TOTP: Time-Based One-Time Password Algorithm <http://tools.ietf.org/html/rfc6238>.

The server then generates a suitable TOTP secret and returns it as a URI, transmitted as the child of the <setup/> element. Note that TOTP MUST NOT be enabled at this point, since it has yet to be tested.

```
<iq type='result' id='123456'>
  <setup xmlns='urn:xmpp:mfa:0'>otpauth://totp/XMPP:portia@venice.
     shakespeare.example?secret=58
     d888c08aa561f370e38cee976121532a883d71&issuer=XMPP</setup>
</iq>
```

Next, the user configures the TOTP Device and generates a code. On the same session, it then completes setup by passing a code:

```
<iq type='set' id='654321'>
  <setup xmlns='urn:xmpp:mfa:0'>123456</setup>
</iq>
```

If the code matches, the server responds with success and TOTP is mandatory for the account from this point.

```
<iq type='result' id='654321'/>
```

6 TOTP SASL2 Tasks

6.1 TOTP-INIT

This task is used to provide (or, more typically, enforce) TOTP enrollment.
This is typically done on first authentication.
There is no initial-response for this task; the server speaks first.
The server sends a challenge containing a TOTP URI. The user should configure their TOTP Device, generate a code, and the client then sends this code to the server as an ASCII string.
If this matches, the Task succeeds, and TOTP is mandatory for the account from this point onward; servers SHOULD NOT require a TOTP task for this SASL2 process however.
There is no additional-data on success or continue with this task.

6.2 TOTP

This task is used to require a TOTP code from the user. In general, this can be one of a group of MFA tasks available to the user, depending on which the user has enrolled for. It MUST NOT be offered to accounts which have not enrolled.
The Task SHOULD NOT be requested if the client has authenticated using CLIENT-KEY, however security concerns might suggest that a Client Key which has not been used for a

lengthy period might benefit from a TOTP challenge.
The client MAY send first using an initial-response.
The server will otherwise send an empty challenge.
The response (or initial-reponse) SHALL be a TOTP code.
If this matches, the Task succeeds.
There is no additional-data on success or continue with this task.

7 Determining Support

Support for the voluntary enrollment protocol by servers is advertised as the Disco feature 'urn:xmpp:mfa:0'.
Support for TOTP itself in client can be determined similarly.

8 Security Considerations

The TOTP secret is a plaintext equivalent shared secret. Both clients and servers MUST protect this. It is RECOMMENDED that it be stored encrypted, with the encryption key held in a distinct location to the per-user TOTP secret. TOTP secrets MUST be hard for an attacker to guess - see RFC 6238 [4] for more detail.

[4] RFC 6238: TOTP: Time-Based One-Time Password Algorithm <http://tools.ietf.org/html/rfc6238>.