Network Working Group
Internet-Draft
Intended status: Informational
Expires: 31 March 2026

B. Contario
Silent Sector

27 September 2025

TOTP Secure Enrollment
draft-contario-totp-secure-enrollment-02

Abstract

This document describes a secure key exchange scheme that extends the Time-Based One-Time Password (TOTP) [RFC6238] de facto enrollment method to prevent compromise of the non-expiring TOTP key by photographic capture of the QR code or by intentional or unintentional persistence of the key string in email, SMS, or other systems outside of the TOTP authenticator system itself.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://Silent-Sector.github.io/totp-secure-enrollment/draft-contario-totp-secure-enrollment.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-contario-totp-secure-enrollment/.

Source for this draft and an issue tracker can be found at https://github.com/Silent-Sector/totp-secure-enrollment.

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 31 March 2026.

Table of Contents

1. Introduction

The common practice of presenting a QR Code for Time-Based One-Time Password (TOTP) enrollment exposes the actual shared TOTP secret key used by the TOTP algorithm for multi-factor authentication. A fallback method of showing the key string to be typed or copied and pasted into a multi-factor authenticator application also exposes the key. In both cases the key that is exposed is the permanent shared TOTP secret key that, by current standards and processes, is expected to never expire.

The below process instead replaces the secret key in the QR code with a URL to the TOTP service for the authenticator application to request the secret key over a secure network transport without authentication. The URL provided uses an atomic one-time use data token (nonce) so only one request for the secret key is honored. If both the user and an attacker use the URL, only one will get the secret key, and only the user will have an active session to complete enrollment by entering the proper TOTP value derived from the secret key. If the attacker gets the secret key, the user will not and the user will be forced to generate a new QR code with a URL to a new secret key until they successfully get the secret key and complete TOTP Verification by entering the current one-time password, meaning the attacker will be guaranteed to not have the same secret key. In cases where the URL to the TOTP service is not accessible, the legacy shared TOTP secret key exchange via QR Code or text string may be offered as a fallback with appropriate warnings displayed against key exposure.

1.1. Terminologies

TOTP: Time-Based One-Time Password as described in [RFC6238], and by itself in this document referring only to the algorithm, not the implementation.

TOTP secret key: the hexadecimal [RFC6238] string representing a value unique to each prover, per [RFC6238] section 3.

hardware token: Usually a dedicated-purpose hardware device that is issued to users. In this document it refers to a device that is pre-set with a TOTP secret key used to display one-time passwords that provide prover functionality.

soft token: Usually a software application that provides enrollment and prover functionality without dedicated hardware, usually running on a smartphone, tablet, personal computer, or password manager application. In this document it refers specifically to an application that stores a TOTP secret key and provides TOTP prover functionality. Sometimes referred to in other documents as a “soft token” or “software token”, as opposed to a “hardware token”.

prover: In this document referring to an implementation of the TOTP protocol that provides a periodically changing one-time password based on the time and a TOTP secret key. A prover may be implemented as an authenticator application (software token) or implemented as a hardware token.

nonce: a randomly-generated cryptographic data token that is used to prevent replay attacks by only allowing the value to be used once, somewhat analogous to a paper ticket that is torn in half or scanned and invalidated when presented for admission to a venue to prevent reuse.

TOTP service: A service used to enroll and verify the TOTP one-time password provided by a prover and associated with a specific authentication request. Also referred to as a “verifier” in [RFC6238], as described below.

verifier: Per [RFC6238] section 3.R1, an authentication or validation server, and in this case for TOTP.

enrollment: also referred to as _provisioning_ per [RFC6238] section 4.2, and _registration_ in [draft-strbac-totp2], the exchange of the TOTP secret key between the verifier and the prover, followed by the verification of the prover’s one-time password by the verifier, and the respective secure storage and linking of TOTP secret key to an identity for MFA.

URL: used explicitly in this document to specify a "Uniform Resource Locator" providing a location and means of access to obtain information via a transfer request. In this document expect the URL to require a transfer using the specified protocol to get the desired information.

otpauth URI: used explicitly in this document to specify a "Uniform Resource Identifier" providing information for immediate use for TOTP enrollment, per [totp-uri-format] and [draft-linuxgemini-otpauth-uri]. In this document expect the otpauth URI to convey the desired information in a single step without an additional transfer step required.

1.2. Requirements Language

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. TOTP Background

The basis of Time-Based One-Time Password (TOTP) [RFC6238] is a symmetric algorithm for both the prover and the verifier to combine a string containing a shared TOTP secret key and the current Unix epoch seconds [RFC3339] which are hashed using SHA-1 or a similar cryptographic hash algorithm. The result is converted to a decimal number and truncated, typically to 6 digits, to be used as the one-time password value that the prover supplies and the verifier accepts as valid only if it calculates the same one-time password value.

TOTP requires that the TOTP secret key, as defined in [RFC6238] section 3 requirement R2, must be known by both the prover and the verifier. The original implementation of TOTP may have been intended for use primarily in hardware tokens that were factory-issued with an unchangeable TOTP secret key, which was a common practice for hardware tokens, since [RFC6238] did not define a protocol for the TOTP secret key exchange. However, increased use of mobile devices that synchronize their time with wireless communication providers and NTP servers have allowed for the proliferation of authenticator applications running on mobile devices, laptops, and desktop computers, all of which require receiving a unique TOTP secret key from each new verifier. Therefore, a protocol evolved as a de facto industry standard for enrollment to exchange the TOTP secret key, described in the next section.

3. TOTP Enrollment

3.1. Overview

For consistency and specificity, the remainder of this document will use the term “TOTP service” in place of “verifier”.

For TOTP enrollment using an authenticator application, it is most logical for the TOTP service to generate the shared TOTP secret key using cryptographically secure random number generators to ensure that each prover has a unique TOTP secret key that has not been used elsewhere. Therefore it is an accepted standard for the TOTP service to generate the key and store it securely, and for the authenticator application to simply accept and store the key in a secure manner.

The current TOTP enrollment protocol was likely influenced or established by the [GoogleAuth] project, which provided a working reference system of two complementary methods for the TOTP service to share the TOTP secret key with the prover during enrollment, described as follows.

3.2. Hexadecimal String Enrollment

The TOTP secret key itself is expressed as a hexadecimal string of an arbitrary length chosen by the TOTP service. During enrollment the TOTP service can display the randomly generated hexadecimal string to the user for them to type or paste into an authenticator application. Typical strengths are >= 160 bits.

3.3. QR Code Enrollment

To avoid typographic errors when the user enters the TOTP secret key into the prover that was provided by the TOTP service, the use of a [QR_Code] was introduced so that manual data entry was not required by the user. The QR Code encodes an otpauth URI scheme, per the [totp-uri-format] and proposed as a draft IETF standard [draft-linuxgemini-otpauth-uri] with the following parameters:

algorithm=(SHA1|SHA256|SHA512)
  OPTIONAL, defaults to SHA1.

secret=[websafe Base32 encoded TOTP secret key, no padding]
  REQUIRED, 128 bits or more.

digits=(6|8)
  OPTIONAL, defaults to 6.

period=N (totp specific)
  OPTIONAL, defaults to 30.

The otpauth URI also includes a human-readable Label to identify the issuer and account name. Some TOTP services also supply a "user" parameter. Both the account name and user parameter could disclose the complete identity that the TOTP secret key applies to. Below is a fictitious example otpauth URI that could be encoded into a QR Code for the user "human@domain.com" using the ExampleCorp service:

otpauth://totp/ExampleCorp%3Ahuman%40domain.com?secret=bl4gbf4ab5l3rj3d6htrxc6bqhrx3m7u&issuer=ExampleCorp

and a urldecoded version of the text:

otpauth://totp/ExampleCorp:human@domain.com?secret=bl4gbf4ab5l3rj3d6htrxc6bqhrx3m7u&issuer=ExampleCorp

Of particular note is that the TOTP secret key value of the "secret" parameter is provided in BASE32 plaintext with no cryptographic protections of any kind, and no expiration or method of revocation for the TOTP secret key if the QR Code or otpauth URI is stored, copied, or transmitted beyond the intended use, or if the TOTP secret key is discovered compromised in any other way.

4. Enrollment Vulnerabilities

The above two enrollment methods do not follow the recommendations listed in [RFC6238] section 5.1 stating:

> All the communications SHOULD take place over a secure channel, e.g., Secure Socket Layer/Transport Layer Security (SSL/TLS) [RFC5246] or IPsec connections [RFC4301].

This results in a wide range of vulnerabilities.

4.1. Hexadecimal String Enrollment Vulnerabilities

Hexadecimal String Enrollment where the TOTP secret key string is visibly presented to the user is often vulnerable to compromise by the following non-exhaustive list of vectors:

1. a nearby observer viewing and remembering the visible TOTP secret key

2. a nearby observer photographing the visible TOTP secret key

3. the user copying the key to the computer’s clipboard and then pasting the TOTP secret key into the prover during enrollment (which exposes the key to potentially malicious applications that monitor the clipboard contents)

4. the user copying the TOTP secret key to the computer’s clipboard and then pasting the TOTP secret key into an email, instant message, or SMS message to send to another user for enrollment (which may persist the TOTP secret key in external or internal systems indefinitely)

5. the user copying the TOTP secret key to a computer’s clipboard and then pasting into a file for later use (which which may persist the TOTP secret key to local or cloud storage)

4.2. QR Code Enrollment Vulnerabilities

Enrollment via [QR_Code] where the encoded TOTP secret key is visible presented to the user is often vulnerable to compromise by the following non-exhaustive list of vectors:

1. a nearby observer photographing the QR Code
2. the user copying the QR Code image into an email or SMS message to send to another user for enrollment (which may persist the TOTP secret key in external or internal systems indefinitely)
3. the user photographing the QR Code image on their mobile device for later use by the authenticator application (which may persist the TOTP secret key in external or internal systems indefinitely)
4. the user photographing the QR Code image on their mobile device to send to another user via email, instant message, cloud service, or SMS for enrollment (which may persist the TOTP secret key in external or internal systems indefinitely)
5. the user printing the QR Code image to complete enrollment later

4.3. Elevated Risk for QR Code Enrollment

As described earlier, the string encoded in the QR Code is a [RFC3986] URI that contains all of the needed information to link the key to the TOTP service and the user with which it is associated, referred to in this document as an otpauth URI. The information provided in the QR Code greatly simplifies the TOTP enrollment since the user does not need to name the entry in their authenticator application or type in a hexadecimal string, so most users opt for the QR Code enrollment method.

However, the tradeoff for this convenience is that the QR Code potentially contains all of the information needed by an attacker to log in to the associated target system as the associated user, except for the user’s first factor password. If the target system is internet-facing and the login page can be identified by name or other intelligence, and the user’s identifier is an email address that can be found in a database of accounts with compromised passwords, then having the TOTP secret key, also in the QR Code, potentially gives the attacker all of the factors needed to log in as the user to the target system.

Because of the vectors by which the QR Code could be unintentionally exposed or stored on systems outside of the user’s control for later harvesting, a secure enrollment method that does not expose the TOTP secret key is needed.

5. TOTP Secure Enrollment

The following method protects the TOTP secret key via HTTPS/TLS as recommended in [RFC6238] section 5.1 when the TOTP service is accessible on a public or private network via a URL that can be generated by the TOTP service.

Since this document is dealing only with TOTP implementations, the remainder of this document will prefer the shortened form of “Secure Enrollment” in place of “TOTP Secure Enrollment”, and “authenticator application” in place of “TOTP authenticator application”

5.1. Mechanism of Operation

The "secret" parameter of the otpauth URI, as described in [totp-uri-format], normally only accepts a BASE32 encoded hexadecimal secret key. However, in a “Secure Enrollment otpauth URI”, the “secret” parameter MUST accept a urlencoded URL with a HTTPS protocol string. This URL, referred to from here forward as the “Secure Enrollment URL”, is how the authenticator application connects to the TOTP service to get a copy of the legacy otpauth URI that contains the secret key.

The [RFC3986] URL specification requires the protocol in a URL be separated by a colon (“:”) which is urlencoded into “%3A”. Because both the “%” and “:” characters are not valid in BASE32 encoding, their presence in the “secret” parameter can effectively be used as a signal that the value of the “secret” parameter should be urldecoded and the resulting value used as the Secure Enrollment URL to request the original format TOTP otpauth URI containing the plaintext TOTP secret key via a network connection.

To minimize the data that is stored in the Secure Enrollment QR code, all of the optional fields SHOULD NOT be included in the Secure Enrollment otpauth URI. Only the “secret” parameter of the otpauth URI should be populated (as shown below).

The actual enrollment data, in the form of an otpauth URI, used by the authenticator application does not change. The enrollment data simply gets delivered via HTTPS instead of embedded in a QR code.

The Secure Enrollment URL MUST include a nonce value that will be marked as invalid after first use by the TOTP service to insure that the request is only ever responded to one time only (not more than once). The Secure Enrollment URL gets added to the "Secure Enrollment otpauth URI", then encoded into a QR Code, then displayed to the user, and then decoded by the user's authenticator application.

A sample Secure Enrollment otpauth URI for ExampleCorp:

otpauth://totp/?secret=https%3A%2F%2Fexamplecorp.com%2Fapi%2Fenrollmfa%2F16062671560671769238465892

The Secure Enrollment URL portion of the URI follows the "?secret=" portion and decodes to:

https://examplecorp.com/api/enrollmfa/16062671560671769238465892

The authenticator application MUST make a HTTP POST request using the Secure Enrollment URL to retrieve the otpauth URI text string in the same legacy format used by authenticator applications prior to adoption of TOTP Secure Enrollment.

It should also be noted that the use of a QR code is not required for Secure Enrollment. The user's authenticator application could accept the Secure Enrollment URL via QR code, copy and paste, or manual entry of the Secure Enrollment URL text itself.

5.2. Device Enrollment Data

TOTP enrollment has historically been a "blind" process because there was no direct interaction between the authenticator application and the TOTP service. Administrators had no information to help end-users that forgot what device or authenticator application they to use to get their 6 digit TOTP codes.

Now the authenticator application makes a HTTP POST request to the TOTP service during Secure Enrollment, so there is an opportunity to send and capture information for administrators to provide hints about which device and authenticator application an end user should use to log in.

When the authenticator application makes the POST request it SHOULD send a request body with device enrollment data in JSON format using Content-Type set to "application/json". The sending of this enrollment data is optional. The recommended content of the JSON data is as follows:

```
{
    "event_type":"totp-secure-enrollment",
    "time_local":"Sun, 22 Sep 2024 23:46:30 -0400",
    "time_utc":"2024-09-23T03:46:30.637Z",
    "device_model":"F990",
    "device_manufacturer":"Foo",
    "os_name":"android",
    "os_version":"12",
    "application_name":"Aegis",
    "application_version":"3.1.1",
    "location_description":"Cincinnati, Ohio, USA",
    "location_longitude":"-84.512",
    "location_latitude":"39.103"
}
```

All element values should be evaluated as strings and contain the following information:

event_type: always the literal string "totp-secure-enrollment" for easy identification by humans and machine

time_local: time of enrollment, recommended formatted as RFC 1123 Date Time for local time of the device for human readability

time_utc: time of enrollment, ISO-8601 instant format recommended for machine readability

device_model: the device model name or the model number

device_manufacturer: the device manufacturer name

os_name: the operating system name

os_version: the operating system version in human readable form

application_name: the authenticator application name or title

application_version: the authenticator application version in human readable form

location_description: if available and permitted, a descriptive name for the location of the device at the time of enrollment, preferably showing country, region / state / province, and municipality

location_longitude: if available and permitted, the coarse longitude of the device at the time of enrollment

location_latitude: if available and permitted, the coarse latitude of the device at the time of enrollment

It is also optional, but recommended, for the TOTP service to store and/or parse the device enrollment data for future use. The data can be used for both end user assistance and administrative review of the different devices being used for TOTP Secure Enrollment.

5.3. Secure Enrollment QR code Workflow

The Secure Enrollment ideal workflow will follow these general steps when using a QR code is viable:

1. *Enrollment Request*: The user initiates the enrollment process for TOTP for 2FA or MFA on their account. The user experience is then passed to the TOTP service.

2. *Secure Enrollment Preparation*: The TOTP service generates a nonce and a random TOTP secret key, storing them both temporarily in a secure location with a configurable validity period (expiration) for human interaction, such as 5 minutes.

3. *Secure Enrollment Presentation*: The TOTP service generates and presents to the user a QR Code that contains an otpauth URI in the format established by [GoogleAuth], but instead of the "secret" parameter containing the TOTP secret key, it contains a urlencoded network-accessible HTTPS URL with only a nonce and no secret key.

4. *User Interaction*: The user interacts with the TOTP authenticator application to scan the QR Code presented.

5. *Secure Key Request*: When the authenticator application detects a “%” or “:” character in the "secret" parameter value of the otpauth URI, the authenticator application performs the new Secure Enrollment functionality by urldecoding the “secret” parameter and using that value in a HTTP POST request.

6. *Secure Key Response*: When the TOTP service receives the HTTP POST request with the nonce, the redeem is atomic. The TOTP service looks up the nonce from temporary secure storage and if the nonce is not expired it atomically expires the nonce and returns the response (over the HTTPS/TLS secure channel) with the otpauth URI containing the secret key in the legacy de facto format, the same data that would normally be provided in the QR Code by the [GoogleAuth] code prior to implementation of Secure Enrollment. At the same time, the TOTP service optionally stores the JSON device enrollment data contained in the POST request body for future use.

7. *TOTP Enrollment*: The authenticator application extracts the "issuer", "secret", "digits", and other parameters from the otpauth URI and registers a new entry in the authenticator application. This step is unchanged from the previous process, except the otpauth URI with the secret key is delivered via HTTPS rather than the QR Code.

8. *TOTP Verification*: The TOTP service presents a user interface that requires the user to enter the current one-time password and verify it before considering the user fully enrolled.

9. *Secure Enrollment Flag*: It is RECOMMENDED that in addition to the new TOTP secret key, the enrollment data stored by the TOTP service should include a Secure Enrollment Flag to indicate that this enrollment was completed over a secure channel. This can be used for auditing purposes and to force users to re-enroll their TOTP authenticator application entries that were not completed in a secure manner in the past.

5.4. Secure Enrollment Copy Paste Workflow

The Secure Enrollment ideal workflow will follow the same general steps as the above QR code workflow when it is easier to copy and paste a text string than to use a QR code, with the exception of the following two steps which are slightly modified and used instead of the same numbered steps: 3. *Secure Enrollment Presentation*: The TOTP service displays in selectable text the otpauth URI in the format established by [GoogleAuth], but instead of the "secret" parameter containing the TOTP secret key, it contains a urlencoded network-accessible HTTPS URL with only a nonce and no actual secret key. The user optionally copies the otpauth URI text into the clipboard or transfers it electronically. 4. *User Interaction*: The user interacts with the authenticator application by pasting or typing in the otpauth URI text and submitting it.

5.5. Workflow Notes

In both the Copy Paste Workflow and the QR code Workflow, the final secret key that is enrolled is never exposed to the user, attackers, or other systems. The secret key is directly exchanged via HTTPS between the TOTP service and the authenticator application.

5.6. Requirements

Secure Enrollment requires the following that is not already defined in [RFC6238]:

1. The TOTP service MUST be able to create a Secure Enrollment URL to an API endpoint that can accept an anonymous HTTPS POST request.

2. The TOTP service API endpoint SHOULD be configured to require non-deprecated TLS ciphers via HTTPS.

3. The Secure Enrollment URL MUST be resolvable and accessible to the authenticator application via internet or private network connection at the time of enrollment.

4. The TOTP service MUST generate a cryptographically secure random string data token to be used as a nonce to both include in the URL and store in secure temporary storage to be accessed by the API endpoint. It is recommended that the nonce have at least 96 bits of entropy. Newly-generated GUID v4 or UUID v4 values with 122 bits of entropy are acceptable nonce values at the time of this writing.

5. The TOTP service MUST set a nonce validity period (expiration) for completion of the human actions. The nonce validity period SHOULD be reasonable for authorized users, such as 5 minutes into the future.

6. The TOTP service MUST NOT permanently store the generated secret key as enrolled until after the user has completed TOTP Verification by entering the current one-time password as a part of an existing user session authenticated with a previous factor.

7. The TOTP service MUST generate an otpauth URI whose secret parameter value is the Secure Enrollment URL, percent-encoded per [RFC3986] section 2. The Secure Enrollment URL MUST NOT include credentials and MUST only include the nonce and necessary routing parameters.

8. The TOTP service MUST exclude optional parts (per [totp-uri-format]) from the Secure Enrollment otpauth URI that is presented to the end user as a QR code or text string to limit data exposure.

9. The TOTP service SHOULD NOT issue http redirects. Redirects MUST NOT downgrade confidentiality (e.g., to http).

10. The authenticator application MUST NOT follow http redirects.

11. The authenticator application MUST allow the otpauth QR code to exclude all optional data from the otpauth URI, including the Label and user/account parameter, only providing the “secret” parameter that is populated with the Secure Enrollment URL.

12. The TOTP service API endpoint MUST invalidate the nonce after its expiration period or atomically redeem it upon first use by the API endpoint, whichever comes first, using appropriate concurrency locking mechanisms to avoid race conditions.

13. The TOTP service MUST generate a new secret key any time a new nonce value is generated for a Secure Enrollment URL. To avoid any race conditions or other security concerns, if the user requests a new QR code or refreshes the page containing the otpauth URI or QR code, a new secret key and nonce value MUST be generated and supplied.

14. The TOTP service API endpoint MUST NOT respond with an actual secret key unless the nonce is found in secure temporary storage, is unused, and the current time is within the validity period. On success, the endpoint MUST return HTTP 200 with a body containing the otpauth URI in the same format established by [GoogleAuth] as described in [totp-uri-format] with the “secret” parameter containing the BASE32 encoded secret key. The response SHOULD include Cache-Control: no-store and SHOULD include Pragma: no-cache. Implementations MUST NOT log the returned secret key or full otpauth URI.

15. The TOTP service API endpoint SHOULD respond with a uniform HTTP 4xx series response error code and message whether the nonce is invalid or expired, preferably a 403 Forbidden response. In environments with specialized security needs the TOTP service API endpoint MAY choose to respond with any other non-usable response, including a TCP disconnect, 200 success code with a blank response, a URL with a decoy TOTP secret key, or any other response that serves the purpose of thwarting an attacker with a false positive response.

16. The TOTP service MAY offer a user-initiated option in the UI to switch to legacy enrollment. Prior to redirecting the user to legacy enrollment the UI SHOULD display a localizable message that warns the user to protect the legacy QR Code and/or the TOTP secret key string from observation and photographic or video capture, and avoid saving, sending, or transmitting them in any form to prevent compromise of the TOTP secret key.

5.7. Reference Code

Pull requests illustrating reference implementations based on working open source code bases can be found here:

TOTP Secure Enrollment authenticator application reference code [totp-secure-enrollment-reference-authenticator]

TOTP Secure Enrollment server framework demo code [totp-secure-enrollment-reference-server]

5.8. Exclusions

TOTP Secure Enrollment does not claim or provide any additional benefit in the following non-exhaustive list of areas:

1. Protection of the secret key once it is stored in the authenticator application after successful Secure Enrollment.

2. If the device containing the authenticator application is obtained by an attacker or is compromised by malware.

3. Assisting in inventorying or tracking enrolled systems to invalidate the enrolled secret key.

4. Purposefully setting up the same TOTP enrollment on multiple authenticator applications for redundancy or shared accounts. This is a usage scenario for TOTP outside of industry best practices and incurs elevated risk. Secure Enrollment is not intended to protect or facilitate this.

5.9. Alternative Network Transports

This specification defines Secure Enrollment only over HTTPS. Use of other transports is out of scope. Implementations MAY define private extensions, but interoperability is not expected.

5.10. Applicability to Other Algorithms

Other multi-factor authentication algorithms / protocols could benefit by adapting this secure enrollment process to exchange shared secrets. The following is a non-exhaustive list of possible candidates:

* HMAC-based One-Time Password (HOTP) [RFC4226]

* OATH Challenge-Response Algorithm (OCRA) [RFC6287]

6. Related Enhancements

There are other efforts to improve the TOTP experience and security, such as proposed in draft [draft-strbac-totp2]. It is believed that the Secure Enrollment method described here can work in conjunction with and enhance the enrollment / registration process as described in [draft-strbac-totp2] section "3. Authentication Scheme Overview" to improve the security of the "TOTP2 Registration" step.

7. Security Considerations

7.1. General

All security considerations for the execution of TOTP and storage of the TOTP secret key are covered in [RFC6238].

7.2. TOTP secret key exchange

The secure exchange of the TOTP secret key MUST use a secure network transport protocol, such as HTTPS. The implementation of the protocol SHOULD follow industry best practices, such as:

1. TOTP service API endpoints SHOULD only allow HTTPS TLS ciphers that are not deprecated by the industry.

2. TOTP service API endpoints and authenticator applications SHOULD negotiate the supported ciphers in preferred order of perceived security by traditional and quantum standards at the time of the exchange, always preferring ciphers that are deemed the most secure against "harvest now, decrypt later" attacks.

3. Clients and servers SHOULD perform certificate validation per the HTTPS PKI; endpoints SHOULD enable HSTS; deployments SHOULD avoid user-modifiable TLS downgrade mechanisms.

7.3. Secret key harvesting

Once it is common knowledge that the legacy QR codes used for TOTP enrollment have no encryption or safeguards, and they can be reused since they contain the non-expiring secret key, attackers may start scanning images in cloud storage or sent via MMS that contain QR codes to harvest the secret key, along with the other details in the QR code otpauth URI. Identification of a target QR code is trivial once decoded into a text string since they all start with “otpauth://” .

7.4. Secure Enrollment Flag

It is RECOMMENDED in the workflow above that in addition to the new TOTP secret key, the enrollment data stored by the TOTP service should include a Secure Enrollment Flag to indicate that this enrollment was completed over a secure channel. It is also RECOMMENDED that action be taken against user accounts that do not have the Secure Enrollment Flag set on their accounts. This action could include, but is not limited to, one or more of the following options that could be happen after a given date for each option:

1. Manual human intervention and verification that the authorized user completes the new Secure Enrollment process.

2. Disable the user accounts until option 1 is performed.

3. When the TOTP service detects success for a user with the Secure Enrollment Flag set to false, fail the login, and send the user through the Secure Enrollment process. Once they complete it, the Secure Enrollment Flag will be set to true and they will be able to log in normally using TOTP authentication.

8. Privacy Considerations

This protocol proposes processing enrollment data (the one-time Secure Enrollment URL/nonce, server-observed timing/network data, and optional client metadata such as device class/OS version and coarse location). Consistent with [RFC6973], servers SHOULD minimize collection and use enrollment data only to complete enrollment and related security functions (e.g., abuse detection); clients SHOULD NOT transmit optional metadata unless the user has clearly opted in or a managed-device policy authorizes it; if location is sent, it SHOULD be coarse (no precise coordinates).

9. IANA Considerations

This document has no IANA actions.

10. References

10.1. Normative References

[GoogleAuth]
"Google Authenticator OpenSource", n.d., <https://github.com/google/google-authenticator>.

[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/rfc/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[RFC6238] M'Raihi, D., Machani, S., Pei, M., and J. Rydell, "TOTP: Time-Based One-Time Password Algorithm", RFC 6238, DOI 10.17487/RFC6238, May 2011, <https://www.rfc-editor.org/rfc/rfc6238>.

[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/rfc/rfc8174>.

[totp-uri-format]
"TOTP URI Format", n.d., <https://github.com/google/google-authenticator/wiki/Key-Uri-Format>.

10.2. Informative References

[draft-linuxgemini-otpauth-uri]
Gemini, L., "The otpauth URI Scheme", I-D draft-linuxgemini-otpauth-uri, August 2023, <https://datatracker.ietf.org/doc/draft-linuxgemini-otpauth-uri/>.

[draft-strbac-totp2]
"TOTP2 Authentication Scheme", n.d., <https://datatracker.ietf.org/submit/status/135578/>.

[QR_Code] "ISO/IEC 18004, currently ISO/IEC 18004:2015 Information technology — Automatic identification and data capture techniques — QR Code bar code symbology specification", n.d., <https://www.iso.org/standard/62021.html>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/rfc/rfc3339>.

[RFC4226] M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D., and O. Ranen, "HOTP: An HMAC-Based One-Time Password Algorithm", RFC 4226, DOI 10.17487/RFC4226, December 2005, <https://www.rfc-editor.org/rfc/rfc4226>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/rfc/rfc4301>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/rfc/rfc5246>.

[RFC6287] M'Raihi, D., Rydell, J., Bajaj, S., Machani, S., and D. Naccache, "OCRA: OATH Challenge-Response Algorithm", RFC 6287, DOI 10.17487/RFC6287, June 2011, <https://www.rfc-editor.org/rfc/rfc6287>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <https://www.rfc-editor.org/rfc/rfc6973>.

[totp-secure-enrollment-reference-authenticator] "TOTP Secure Enrollment reference authenticator application pull request", n.d., <https://github.com/silentbrianc/Aegis/pull/1/>.

[totp-secure-enrollment-reference-server] "TOTP Secure Enrollment reference server pull request", n.d., <https://github.com/silentbrianc/aspnetcore/pull/2/>.

Acknowledgments

Thank you to Lauro Chavez and Zach Fuller at Silent Sector for their reviews, suggestions, and support.

Thank you to Al Katawazi and Craig Tulli of Blueshift Technologies for reviews and feedback.

Thank you to Mike Amundsen of Amundsen.com for guidance in starting the IETF draft process.

Author’s Address

Brian Contario
Silent Sector
Email: bcontario@silentsector.com