

Two-Factor-Authentication (2FA)

von Marc-Niclas Harm, Kryptologie, TH-Lübeck

Was ist 2FA ?

Die **Two-Factor-Authentication (2FA)** ist eine Unterform der **Multi-Factor-Authentication (MFA)**. Der Hauptzweck der **MFA** liegt darin, einen Benutzer zu **identifizieren** und/oder zu **authentisieren**. Dabei müssen **mindestens zwei verschiedene** der folgenden Faktoren verwendet werden:

Faktor	Beispiel
Wissen 	Passphrase wie <i>Passwort, PIN</i>
Besitz 	Security-Token wie <i>USB-Token, Chip-Karte</i>
Inhärenz 	biometrische Charakteristika wie <i>Fingerabdruck, Unterschrift</i>

Bei **2FA** sind es genau **zwei verschiedene** Faktoren, welche gegeben sein müssen. Bei **Applikationen im Web** sind diese **zwei** Faktoren überwiegend das **Wissen** in Form eines Passworts und der **Besitz** in Form eines **Software-Tokens**.

Wieso ist 2FA heutzutage wichtig/notwendig ?

Immer häufiger liest man im Internet oder in der Zeitung, dass beim Unternehmen XYZ tausende persönliche Daten **gestohlen** wurden. Egal, ob verursacht durch immer **anspruchsvoller werdende Kriminelle** oder durch ein **einfaches Datenleck**, am Ende ist es auch der Nutzer, der leidet.

Falls nun der unwahrscheinliche Fall eintritt, dass diejenigen, die die Daten in die Hände bekommen haben, es schaffen die **Passwörter** der Nutzer zu "entschlüsseln" (liegen meistens in *Hash-Form* vor), dann haben all jene Nutzer dieser Menge ein Problem, welche dieses **Passwort** noch bei anderen **Diensten** nutzen und dort keine **2FA** aktiviert haben. Der **Zugriff** auf diese **Accounts** ist nun ein Leichtes.

Man kommt somit zu dem Schluss, dass **Passwörter** alleine heutzutage **nicht mehr** zum Schutz beim **Login** von Diensten **ausreichen** und ein **zusätzlicher Schutz** wie die **2FA** nötig sind.

Zwei gängige Verfahren der 2FA: HOTP und TOTP

HOTP (RFC 4226 aus dem Jahr 2005)

HMAC-Based One-Time Password

$$\$HOTP = \text{Truncate}(\text{HMAC} - \text{SHA} - 1(K, C))\$$$

Name	Beschreibung
K	Schlüssel
C	Zähler
HMAC	Keyed-Hash Message Authentication Code
SHA-1	Secure Hash Algorithm 1
Truncate	Konvertiert Hash in HOTP

Nachteile von HOTP

Wenn man die **HOTPs** beispielsweise von zwei verschiedenen Geräten aus generiert, kann es vorkommen, dass die **Zähler** der beiden Geräte **asynchron** werden. Hier muss dann eine Möglichkeit gefunden werden, die **Zähler zu synchronisieren**.

Weiterhin ist ein generiertes und anschließend benutztes **HOTP** solange gültig, bis ein Weiteres generiert, benutzt und der **Zähler** auf der **Serverseite** erhöht wurde.

Ferner können **HOTPs** mittels der **Brute-Force-Methode** (Ausprobieren aller möglichen Werte) gefunden werden. Dies muss mit Hilfe einer **Sperrung der Eingabe von HOTPs** nach einer bestimmten Anzahl an Fehlversuchen für ein bestimmtes Zeitintervall unterbunden werden.

TOTP (RFC 6238 aus dem Jahr 2011)

Time-Based One-Time Password Algorithm

$$\$TOTP = \text{HOTP}(K, T) \$ \$T = \text{Floor}((\text{Unixtime}(Now) - \text{Unixtime}(T0)) / T1)\$$$

Name	Beschreibung
K	Schlüssel
Now	Aktuelles Datum & Zeit
T0	1. Januar 1970, 00:00 Uhr UTC (Start der Unixzeit)
T1	Gültigkeitsintervall
Unixtime	Konvertiert Datum & Zeit in Unix-Zeitstempel
Floor	Rundet auf die nächste ganze Zahl ab

Vorteile von TOTP

Dadurch, dass der **Zähler** nun durch einen **Unix-Zeitstempel** repräsentiert wird, ist die **Synchronisation** von **Zählern** nicht mehr nötig, unter der Bedingung das **Client** und **Server** den aktuellen **Unix-Zeitstempel** abrufen können.

Weitere 2FA Möglichkeiten

SMS, Anruf, E-Mail

Die drei Möglichkeiten **SMS**, **Anruf** und **E-Mail** basieren alle darauf, dass einem nach Eingabe der **Telefonnummer/E-Mail-Adresse** das **OTP** zugesendet wird (sei es durch Text oder durch Ton). Das Problem dabei ist, dass diese im eigentlichen Sinne nicht für den Einsatz als Verfahren der **2FA** gedacht waren. **Telefonnummern** und **E-Mail-Adressen** können neu vergeben, und **SIM-Karten** gehackt werden.

Security-Token

Bei einem **Security-Token** wird eine **Hardwarekomponente** benutzt, um sich zu **identifizieren/authentifizieren**. Ein Beispiel dafür sind **U2F-Geräte** mit dem **U2F-Standard**.

U2F-Standard

Universal Second Factor

Beim **U2F-Standard** wird zur **Authentifizierung** eine Art des **Challenge-Response-Verfahrens** angewandt, welches - knapp formuliert - wie folgt funktioniert:

1. Nach Prüfung von z.B. **Nutzernamen** und **Passwort** kommt es zur **2FA** via **U2F** und der **Benutzer** schließt sein **U2F-Gerät** - beispielsweise über **USB** - an sein **Endgerät** an
 2. Der **Server** schickt eine **Challenge** und eine **Schlüsselkennung**
 3. Der **Client** (z.B. im Web der Browser) leitet die Daten an das **U2F-Gerät** weiter
 4. Der **Benutzer** muss den Vorgang nun bestätigen (z.B. durch einen Knopf auf dem U2F-Gerät). Nach der Bestätigung nutzt dann das **U2F-Gerät** den zur **Schlüsselkennung** passenden **privaten Schlüssel**, um damit die **Challenge** zu **signieren**. Die erstellte **Signatur** wird danach zurück an den **Browser** geleitet.
 5. Der **Browser** schickt die **Challenge** mit der **Signatur** zurück an den **Server**, welcher mithilfe des zur **Schlüsselkennung** passenden **öffentlichen Schlüssels** die Signatur prüft und anhand des **Ergebnisses** den Zugang entweder **gewährt** oder **nicht gewährt**.
- (Setzt voraus, dass das U2F-Gerät des Benutzers und der Server vorher ein Schlüsselpaar ausgehandelt haben.)

GitHub

Repository: <https://github.com/kodehat/kryptologie-2fa>

Quellen

- <https://authy.com/what-is-2fa/>
- <https://itsecblog.de/2fa-zwei-faktor-authentifizierung-mit-totp/>
- <https://fidoalliance.org/specs/fido-u2f-v1.0-rd-20140209/fido-u2f-overview-v1.0-rd-20140209.pdf>
- https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/m/m04/m04133.html
- <https://digitalguardian.com/blog/uncovering-password-habits-are-users-password-security-habits-improving-infographic>
- <https://tools.ietf.org/html/rfc4226>
- <https://tools.ietf.org/html/rfc6238>