'twofac' Secret Store

Software Requirements Specification

Inhaltsverzeichnis

1 Einleitung

1.1 Zweck

Dieses Dokument stellt eine detaillierte Beschreibung der Anforderungen der 'Secret Store' Software. Es beschreibt die geforderten Funktionen im Kontext, und beinhaltet zudem einen Implementationsplan.

1.2 Umfang

Der 'Secret Store' ist das Rückgrat welches das sichere Speichern und Abfragen von 'Secrets' in und von einer Datenbank durch die 'twofac' Benutzeroberfläche ermöglicht.

Demzufolge befindet sich 'Secret Store' auf einem Web-Server und kommuniziert mit einer Datenbank.

1.3 Begriffserklärungen und Abkürzungen

Ausdruck	Definition
[Shared] Secret	Geheim zu haltende Zeichenkette, verwendet um zeitlich limitierte Einmalpasswörter zu generieren
`twofac`	Web-Implementation des TOTP Algorithmus unter besonderer Berücksichtigung des von der Plattform Steam verwendeten Formats, konsumiert Secrets
TOTP	Time-based One-time Password Algorithmus, Verwendung siehe Secret
Store	Abk. Secret Store
Node.js	Serverseitige Plattform in der Softwareentwicklung zum Betrieb von Netzwerkanwendungen (i. e. Webserver).
Back-End	Im Ggs. zu 'Front-End' (vgl. `twofac`); nahe an der Datenverarbeitung, weit weg von der Eingabe
TLS	Transport Layer Security, Verschlüsselungsprotokoll zur Datenübertragung im Internet
Account	Sammlung von Daten unter einem eindeutigen Namen
CPRNG	Kryptographisch sicherer (pseudo) Zufallszahlengenerator
$X = Hash(Y)$	Bildet eine beliebige Eingabemenge Y auf eine Zielmenge fester Länge X ab. Von X können keine Rückschlüsse auf Y gemacht werden.
$X = Encrypt(Y, K_Y)$	Verschlüsselt eine beliebige Eingabemenge Y mit dem Schlüssel K_Y.
$Y = Decrypt(X, K_Y)$	Entschlüsselt eine beliebige Eingabemenge Y mit dem Schlüssel K_Y.

2 Allgemeine Beschreibung

2.1 Produktperspektive

Der Secret Store hängt eng mit der bestehenden `twofac` Webpräsenz zusammen und ist ausschließlich durch die von eben jener gegebenen Benutzeroberfläche ansprech- und abfragbar.

2.2 Produktfunktionen

Die vom Store gegebenen Schnittstellen erlauben das sichere und (aus der Sicht des Endbenutzers) unkomplizierte Speichern und Abfragen von sicherheitskritischen Secrets.

2.3 Benutzermerkmale

Keine Erfahrung bzgl. Kryptographie o.ä. ist vorausgesetzt oder erwartet.

3 Spezifische Anforderungen

Hauptbestandteil ist das in Node.js implementierte Secret Store Back-End, welches alle öffentlich zugänglichen Schnittstellen bereitstellt. Peripher befindet sich eine nicht zwingend kryptographisch gesicherte Datenbank.

Das Back-End ist mit der **benutzerspezifischen Zugangskontrolle** zu den bereitgestellten Schnittstellen und mit der **Aufbereitung und Ausführung des Datenverkehrs** zwischen sich selbst und der Datenbank betraut. Genauer umfasst dies unter anderem auch die **kryptographische Ver- und Entschlüsselung** der zu sichernden oder gesicherten Secrets.

Im folgenden werden alle Abläufe dokumentiert. Es wird angenommen, dass die `twofac`-Benutzeroberfläche (welche außerhalb des hier gegebenen Rahmens ist) angemessen angepasst wurde und jegliche Kommunikation über das TLS Protokoll abgewickelt wird.

3.1 Sign-Up und Sign-In

Ein den Account identifizierender *Username* U und ein accountspezifisches *Passwort* P werden an den Store übermittelt.

0. Existiert ein Account U wird ein *Sign-In*, andernfalls ein *Sign-Up*, initiiert.

3.1.1 Sign-Up (`POST /signup`)

1. Zwei Zeichenketten (Salts) aus mithilfe eines CPRNG zufällig gewählten Zeichen S_P und S_K werden generiert.
2. Der Hash $H_P = Hash(P + S_P)$ wird generiert.
3. U, H_P, S_P und S_K werden in der Datenbank gespeichert.
4. Ein signierte und in einer angemessenen Zeitspanne ablaufender JWT T_U wird übermittelt.

3.1.2 Sign-In (`POST /signin`)

1. Die U zugehörigen Felder H_P und S_P werden von der Datenbank abgerufen.
2. Die Wahrheit der Gleichung $H_P = Hash(P + S_P)$ wird überprüft – ist sie nicht gegeben wird der Vorgang abgebrochen.
3. Ein signierte und in einer angemessenen Zeitspanne ablaufender JWT T_U wird übermittelt.

3.2 Hinzufügen eines Secrets (`POST /secret/add`)

Ein JWT T_U, ein *Alias* A, ein *Secret* S_{plain} und das zum Account U gehörende *Passwort* P werden an den Store übermittelt.

0. Ist
   - T_U ungültig,
   - A in der Datenbank nicht eindeutig oder
   - $H_P \neq Hash(P + S_P)$, wobei H_P und S_P aus einer Datenbankabfrage hervorgehen und U zugehörig sind,

   wird der Vorgang abgebrochen.

1. Das U zugehörige Feld S_K wird von der Datenbank abgerufen.
2. Der Schlüssel $K = Hash(P + S_K)$ wird generiert.
3. Das Secret $S_{crypt} = Encrypt(S_{plain}, K)$ wird generiert.
4. S_{crypt} wird in der Datenbank gespeichert.

3.3 Auslesen eines Secrets (`GET /secret/get?alias=`)

Ein JWT T_U, ein *Alias* A und das zum Account U gehörende *Passwort* P werden an `secstore` übermittelt.

0. Ist
   - T_U ungültig,
   - A nicht in der Datenbank enthalten oder
   - $H_P \neq Hash(P + S_P)$, wobei H_P und S_P aus einer Datenbankabfrage hervorgehen und U zugehörig sind,

   wird der Vorgang abgebrochen.
1. Das U zugehörige Feld S_K und das $A \in U$ zugehörige Feld S_{crypt} wird abgerufen.
2. Der Schlüssel $K = Hash(P + S_K)$ wird generiert.
3. Das Klartext-Secret $S_{plain} = Decrypt(S_{crypt}, K)$ wird generiert.
4. S_{plain} wird übermittelt