Authentifizierung
Projekt BierIdee

Danilo Bargen, Christian Fässler, Jonas Furrer

16. April 2012

Inhaltsverzeichnis

Änderungshistorie

Version	Datum	Änderung	Person
v1.0	14.04.2012	Dokument erstellt	cfaessle
v1.1	16.04.2012	Review	dbargen

1 Einleitung

Der Zweck dieses Dokumentes ist das Erläutern der verwendeten Authentifizierungsmechanismen für die REST API im Projekt Bieridee. Eine wichtige Voraussetzung ist, dass auch dieser Aspekt *RESTful* implementiert ist, wobei vor allem das Prinzip der *Statelessness* zum Tragen kommt.

2 Verfahren

Zur Authentifizierung kommt das HMAC[1] Verfahren gemäss RFC 2104[2] zum Einsatz. Alle Requests (Messages) werden mittels Passwort und einer Hashfunktion signiert.

2.1 Registrierung

Der Benutzer registriert einen neuen Benutzeraccount. Zum Benutzeraccount gehört ein Passwort, welches dem Server hashed (SHA-256) übermittelt wird. Der Hashwert wird nachfolgend „Secret“ genannt.

2.2 HTTP Request vorbereiten

Der Client bereitet seinen HTTP Request vor.

2.3 Signatur berechnen

Über ein definiertes Set der HTTP Request Attribute wird eine Signatur (Hashwert) gebildet. Für die Bildung dieses Hashwertes wird gemäss RFC 2104 das Secret verwendet.

Für das genaue Berechnungsverfahren des HMAC, siehe Anhang *5.1*.

2.4 Authorization Header setzen

Der Request wird mit einem zusätzlichen HTTP Header versehen, welcher den Benutzernamen sowie die generierte Signatur beinhaltet:

```
Authorization: <username>:<signatur>
```

Anschliessend wird der Request abgesendet.

[1] Hash-based Message Authentication Code, siehe http://en.wikipedia.org/wiki/HMAC
[2] http://www.ietf.org/rfc/rfc2104.txt

2.5 Signatur verifizieren

Der Server empfängt den Request und extrahiert den `Authorization` Header. Anhand des Benutzernamens kann er in der Datenbank das gemeinsame Secret ausfindig machen und über das selbe Set von Attributen auch die Signatur berechnen.

2.6 Benutzer authentifizieren

Die beiden Signaturen werden verglichen. Stimmen Sie überein, ist der User erfolgreich authentifiziert.

3 Technische Details

3.1 Hashing Algorithmus

Als Hash Algorithmus wird SHA-256[3] verwendet.

3.2 Speicherung der Passwörter

Die Passwörter werden – sowohl in der Datenbank wie auch im Client – ebenfalls hashed gespeichert. Somit haben alle Passwörter ein einheitliches Format. Keine Sonderzeichen, gleiche Länge. Für die HMAC Bildung werden die Hashwerte dieser Passwörter verwendet.

4 Sicherheit

4.1 Aushandlung gemeinsames Secret

Der einzige Schwachpunkt der angedachten Methode ist der initiale Austausch des gemeinsamen Secrets.

4.2 Hashing Algorithmus

Der SHA-256 Algorithmus ist ein standardisierter und weit verbreiteter Hashing-Algorithmus. Er bietet mehr Sicherheit als MD5 und SHA-1 und wird desweiteren von Java out-of-the-box unterstützt.

[3] http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf

5 Anhang

5.1 Berechnung des HMAC

Der HMAC wird aus der Nachricht N und einem geheimen Schlüssel K mittels der Hash-Funktion H nach RFC 2104 wie folgt berechnet. K wird auf die Blocklänge B der Hash-Funktion (512 Bit für die meisten gängigen Hash-Funktionen) aufgefüllt. Falls die Länge von K grösser als die Blocklänge der Hash-Funktion ist, wird K durch $\mathrm{H}(K)$ ersetzt.

$$\mathrm{HMAC}_K(N) = \mathrm{H}\left((K \oplus opad) \;||\; \mathrm{H}\left((K \oplus ipad) \;||\; N\right)\right)$$

Die Werte *opad* und *ipad* sind dabei Konstanten, $\oplus$ steht für die bitweise XOR-Operation und $||$ für die Verknüpfung durch einfaches Zusammensetzen (Konkatenation).

Nach RFC 2104 sind beide Konstanten wie folgt definiert:

$$opad = \underbrace{0\text{x}5\text{C}\ldots0\text{x}5\text{C}}_{B\text{-mal}} \text{ und } ipad = \underbrace{0\text{x}36\ldots0\text{x}36}_{B\text{-mal}}$$