KvBK: Authentifizierung und Autorisierung im Web: BasicAuth, DigestAuth, OAuth

Julian Reisser

21. Juli 2010

Inhaltsverzeichnis

1 Authentifizierung und Autorisierung im Web

Das Web ist die meistgenutzte Komponente des Internets. Basierend auf HTTP, dem Hypertext Transfer Protokoll entwickelte sich das World Wide Web innerhalb von Jahren derart, dass es heute nicht mehr wegzudenken ist. Unzählige Webseiten bieten öffentlich zugängliche oder private, kostenfreie oder kostenpflichtige Dienste und Informationen an. Eine Einschränkung der Benutzergruppe durch den Anbieter des Inhalts ist deshalb unverzichtbar. Ohne die Möglichkeit im Web Personen zu identifizieren und ihnen Rechte einzuräumen wären viele Inhalte des Webs nicht machbar. Authentifizierung und Autorisierung spielen deshalb eine fundamentale Rolle.

2 Basic Authentication

Basic Authentication ist die Standardimplementierung der Authentifizierung im Web. Es ist Teil des Hypertext Transfer Protokolls „HTTP/1.0“. Die Funktionsweise ist simpel: Mit dem HTTP-Statuscode 401 kann ein Webserver dem Benutzer signalisieren, dass die angeforderte Ressource passwortgeschützt ist. Zusätzlich befindet sich eine `WWW-Authenticate` Headerzeile in der Antwort, die den Namen des geschützten Bereichs (`realm`) beinhaltet [Abb. 1].

```
401 Unauthorized
WWW-Authenticate: Basic realm = "some protected area"
[...]
```

Abbildung 1: Antwort des Webservers wenn Benutzer geschützte Ressource anfordert.

Der Benutzer kann anschliessend seine Anfrage mit der zusätzlichen `Authorization` Headerzeile wiederholen, in der Benutzername und Passwort durch einen Doppelpunkt getrennt Base64-kodiert sind [Abb. 2].

```
GET /get/index
Authorization: Basic S3ZCSzpLdkJL
[...]
```

Abbildung 2: Anfrage des Benutzers mit Benutzername und Passwort: „KvBK:KvBK“ Base64-kodiert.

Da die Zugangsdaten lediglich kodiert, nicht aber verschlüsselt werden, können sie wenn sie mitgehört werden leicht dekodiert werden. Eine sichere Form der Authentifizierung bietet Basic Authentication deshalb nur zusammen mit einer verschlüsselten Verbindung. [1]

3 Digest Authentication

Digest Authentication, der Nachfolger von Basic Authentication wurde entworfen um die Sicherheitsmängel des Vorgängers zu beseitigen. Die Grundidee besteht darin, dass die Benutzerdaten nicht mehr direkt übermittelt werden, sondern nur noch Benutzername und ein Hashwert bestehend aus dem Passwort, einer Zufallszahl des Webservers und ggf. anderen Variablen übertragen werden. Der Webserver kann dann seinerseits den Hashwert berechnen und ihn mit dem Empfangenen vergleichen.

Die WWW-Authenticate Headerzeile die der Webserver dem Benutzer schickt ist dementsprechend um einige Parameter erweitert [Abb. 3].
`realm`: Name des geschützten Bereichs.
`domain`: Liste von URIs für welche der Zugriff gilt.
`nonce`: Zufallszahl des Webservers, gebildet mithilfe eines Zeitstempels um die Gültigkeit der `nonce` zeitlich zu beschränken und somit Replay-Angriffe zu verhindern.
`algorithm`: Vorgeschlagener Algorithmus um den Hashwert zu bilden; Standardalgorithmus: „MD5“.
`qop-options`: qop steht für „quality of protection“. Dem Benutzer wird mitgeteilt welche zusätzlichen Sicherheitsmechanism der Webserver unterstützt.

```
WWW-Authenticate: Digest
realm = "",
domain = "",
nonce = "",
algorithm = "",
qop-options = "",
[...]
```

Abbildung 3: Die WWW-Authenticate Headerzeile bei Digest Authentication.

Auch die Authorization Headerzeile wurde erweitert [Abb. 4].
`username`: Benutzername.
`response`: Der Hashwert aus Passwort, `nonce` und mehr.
`nonce`: Die `nonce` des Webservers.
`algorithm`: Gewählter Algorithmus.
`message-qop`: Gewählte qop.
`nonce-count`: Anzahl an Paketen die mit dieser `nonce` bist jetzt angefordert wurden. Es handelt sich um einen zusätzlichen Sicherheitsmechanismus der durch entsprechendes Setzen von `message-qop` aktiviert werden kann und dient der einfachen Erkennung von Replay-Angriffen.
`cnonce`: Eine vom Benutzer generierte Zufallszahl. Kann durch Setzen von `message-qop` aktiviert werden und dient der Erkennung von Chosen-Plaintext-Angriffen.

```
Authorization: Digest
username = "",
response = "",
nonce = "",
algorithm = "",
message-qop = "",
nonce-count = "",
cnonce = "",
[...]
```

Abbildung 4: Die Authorization Headerzeile bei Digest Authentication.

Der Hashwert wird gebildet, indem Benutzername, Passwort, ein Doppelpunkt, und `nonce` in einem String konkateniert werden. Je nachdem ob und welche `message-qop` gewählt wurde, fliesst ggf. auch `message-qop`, `cnonce` und `nonce-count` in den String ein. Anschliessend wird der String mit dem gewählten Algorithmus `algorithm` gehasht [Abb. 5].

```
algorithm( concat(username, password, [...], ":", nonce, [...]) )
```

Abbildung 5: Bilden des Hashwerts aus Benutzername, Passwort, `cnonce` (und mehr).

Wenn der Webserver eine Anforderung des Benutzers mit den richtigen Benutzerdaten akzeptiert, wird ebenfalls eine eine Authentication-Info Headerzeile angefügt, welche die `nonce` für die nächste Anfrage vorgibt. Ob der Webserver wirklich nach jeder Anfrage die `nonce` ändert ist dem Server überlassen [Abb. 6]. In der Praxis wird meist darauf verzichtet um dem Benutzer zu erlauben mehrere Dateien mit einer Anfrage zu holen (Pipelining). [1]

```
Authentication-Info
nextnonce = "",
[...]
```

Abbildung 6: Der Webserver teilt dem Benutzer in der Authentication-Info Headerzeile die `nonce` für die nächste Anfrage mit.

4 OAuth

OAuth ist ein offenes Protokoll, welches sichere Autorisierung über HTTP ermöglicht. Es bietet dem Nutzer die Möglichkeit einem Dienst zu erlauben auf seine Daten zuzugreifen, die durch einen anderen Dienst bereitgestellt werden, ohne seine Benutzerdaten weiterzugeben. Die erste Version erschien 2007 von Blaine Cook und Chris Messina.

Häufig gebrauchte Begriffe im Zusammenhang mit OAuth:
`Service Provider`: HTTP Server der Ressourcen für `Konsumenten` bereitstellt.
`Konsument`: Dienst der auf Ressourcen von `Service Providern` zugreift.
`Anfragetoken`: Ein Token, das nachdem es vom Endbenutzer verifiziert wurde gegen ein `Zugriffstoken` getauscht werden kann.
`Zugriffstoken`: Ermöglicht Zugriff auf Ressourcen eines Service Providers.

Bevor das System für den Endbenutzer funktioniert, muss der `Service Provider` dem `Konsumenten` einmalig einen `Konsumentenschlüssel`, ein `Konsumentengeheimniss` und drei URLs mitteilen. Der `Konsumentenschlüssel` und das `Konsumentengeheimniss` dienen der Authentifizierung des Konsumenten gegenüber dem Service Provider. Dabei geschieht jeder Vergleich von Geheimnissen mit dem Service Provider analog zu Digest Authentication, d.h. das `Konsumentengeheimniss` wird ausschliesslich gehasht (zusammen mit anderen Variablen) übertragen. Unter den drei URLs kann sich der `Konsument` `Anfragetoken` und `Zugriffstoken` beschaffen, und den Endbenutzer zur Verifizierung eines `Anfragetokens` weiterleiten. [2]

Das weitere Prinzip wird im Folgenden an einem Beispiel veranschaulicht [Abb. 7]. Im Beispiel will der Endbenutzer Photos die er bei Google Picasa (`Service Provider`) gespeichert hat von einem Druckdienst (`Konsument`) drucken lassen. Dazu wählt er auf der Webseite des Druckdienstes die Option Bilder von Google Picasa zu drucken (1). Daraufhin fordert der Druckdienst ein `Anfragetoken` von Google Picasa (2) und bekommt es (3). Wie diese Anfrage im genauen aussieht illustrieren [Abb. 8] und [Abb. 9]. Als nächstes muss der Endbenutzer zur Verifikation des `Anfragetokens` an Google Picasa weitergeleitet werden (4/5). Die dafür benötigte URL wird gebildet mit Hilfe der einmalig vereinbarten URL zwischen dem Druckdienst und Google Picasa und dem erhaltenen Anfragetoken. Dort kann der Nutzer durch Eingabe seiner Benutzerdaten bestätigen, dass der Druckdienst die Erlaubnis hat auf seine Photos zuzugreifen (6). Nachdem die Bestätigung erfolgt ist, wird der Endbenutzer zurück zum Druckdienst geleitet (7). Das `Anfragetoken` kann nun gegen ein `Zugriffstoken` ausgetauscht werden (8/9), wie in [Abb. 10] und [Abb. 11] gezeigt wird. Nun kann der Druckdienst die Photos anfordern und laden (10/11), siehe [Abb. 12]. [3]

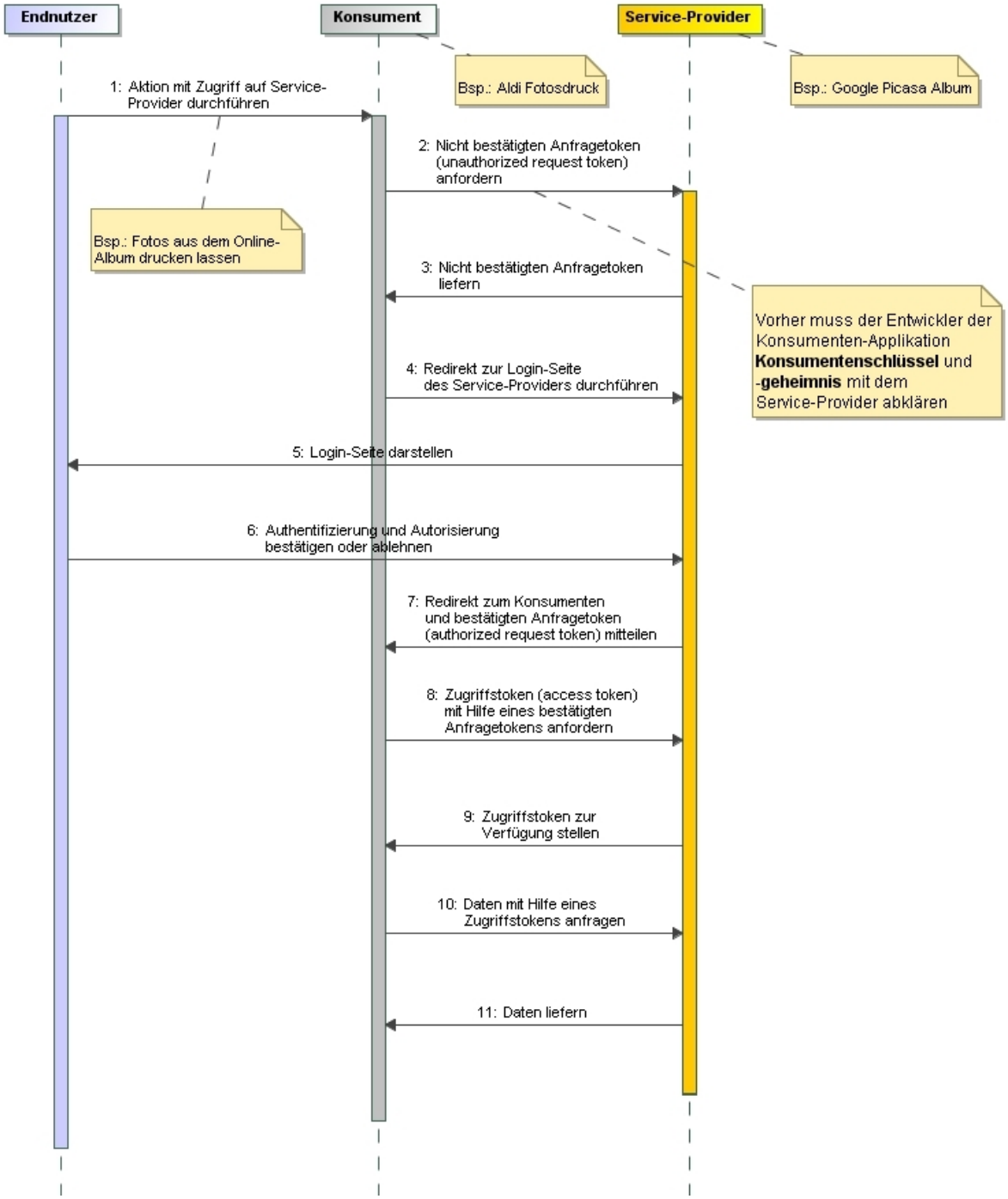

Abbildung 7: Veranschaulichung von OAuth an einem Beispiel. [4]

```
POST /anfragetoken HTTP/1.1
Host: picasa.google.de
Authorization: OAuth
oauth_consumer_key = "",
oauth_signature = "",
[...]
```

Abbildung 8: Das `Anfragetoken` wird angefordert unter der Beispieladresse: picasa.google.de/anfragetoken. Die `oauth_signature` ist ein Hashwert aus `Konsumentenschlüssel` (und anderen Variablen), analog zu Digest Authentication.

```
HTTP/1.1 200 OK
Content-Type: application/x-www-form-urlencoded
oauth_token = "",
oauth_token_secret = "",
[...]
```

Abbildung 9: Der `Konsument` (Bsp. Druckdienst), bekommt das `Anfragetoken`.

```
POST /zugriffstoken HTTP/1.1
Host: picasa.google.de
Authorization: OAuth
oauth_consumer_key = "",
oauth_token = "",
oauth_signature = "",
[...]
```

Abbildung 10: Anfragen des `Zugriffstokens` unter der Beispieladresse: picasa.google.de/zugriffstoken. `oauth_token` ist das `Anfragetoken`, `oauth_signature` der Hashwert aus `oauth_token_secret` (und anderen Variablen).

```
HTTP/1.1 200 OK
Content-Type: application/x-www-form-urlencoded
oauth_token = "",
oauth_token_secret = "",
[...]
```

Abbildung 11: Der `Konsument` (Bsp. Druckdienst), bekommt das `Zugriffstoken`.

```
GET /photos
Host: picasa.google.de
Authorization: OAuth
oauth_consumer_key = "",
oauth_token = "",
oauth_signature = "",
[...]
```

Abbildung 12: Der `Konsument` (Bsp. Druckdienst), fordert Datein des `Service Providers` (Bsp. Google Picasa) an. `oauth_signature` ist der Hashwert aus `oauth_token_secret` (und anderen Variablen).

5 Zusammenfassung

Es wurde gezeigt, dass mit Basic Authentication ein simpler aber einfach zu benutzender Standard für das Authentifizieren über das Web geschaffen wurde.
Digest Authentication schliesslich beseitigte die Sicherheitsmängel seines Vorgängers und bietet eine solide, sichere Methode Resourcen zu schützen, ohne dem Entwickler der Website einen grossen Mehraufwand abzuverlangen.
Eine mögliche Lösung des grundlegenden Problems einem Dienst Zugriff auf Ressourcen eines anderen Dienstes zu verschaffen, ohne die Benutzerdaten an Dritte weitergeben zu müssen, wurde mit dem OAuth Protokoll aufgezeigt.

Literatur

[1] RFC 2617: HTTP Authentication: Basic and Digest Access Authentication: http://tools.ietf.org/html/rfc2617

[2] Heise Artikel: Herr Müller, sind Sie's? : http://www.heise.de/developer/artikel/Autorisierungsdienst mit-OAuth-845382.html

[3] RFC 5849: The OAuth 1.0 Protocol: http://tools.ietf.org/html/rfc5849

[4] [BILD] Heise Artikel: Herr Müller, sind Sie's: http://www.heise.de/developer/artikel/Szenarien-und-Spezifikationen-845431.html?view=zoom;zoom=4