Übungsblatt 02

Thomas Samy Dafir, Lex Winandy

Aufgabe 1

Gruppe: Konfigurieren Sie im Webserver einen VirtualHost, der auf port 8765 lauscht. Erstellen Sie ein separates DocumentRoot für diesen VirtualHost.

Zum Konfigurieren eines VirtualHost muss man in /etc/apache2/sites-available/ eine neue .conf Datei erstellen. In dieser Datei soll nun der VirtualHost eingetragen werden und auf welchem Port er lauscht.
*<VirtualHost *:8765> ... </VirtualHost>* ⇒ Angabe auf welchen Port er reagieren soll
DocumentRoot /var/www/virtualHost/ ⇒ Angabe wo das Basisverzeichnis liegt
Für das Verzeichnis müssen passende Berechtigungen gesetzt werden: access + read für Ordner, read für Dateien.
sudo a2ensite virtualHost.conf ⇒ VirtualHost aktivieren

Aufgabe 2

Erstellen Sie eine passwortgeschützte URL. Verwenden Sie digest authentication. Was ist der Unterschied zu plain. Hinweis: .htaccess, mod auth digest.

Funktion: Der Client sendet eine Anfrage an den Server. Dieser antwortet mit "not authorized" und sendet einen einmal gültigen Zufallswert mit. Der Client wird zur Passworteingabe aufgefordert. Der User authentufuziert sich. Der Client sendet einen neuen Request, jetzt aber mit einem zusätzlichen Header: WWW_Authenticate. Enthält User, realm, qop, den Zufallswert und das gehashte Passwort: MD5 :(. Der Server vergleicht mit den Daten im Passwortfile und sendet entwerde 200 OK oder wiederum 401 not authorized.
Unterschied zu basic: password wird gehashed übertragen.
Benötigte Module: mod_auth_digest + mod_authn_file, für User/Password file.
Verwendung:

1. Optionen für Modul spezifizieren (2 Möglichkeiten):
    - Über Directory Eintrag in der Apache config
    - Über eine .htaccess Datei im zu schützenden Verzeichnis

2. Datei mit "user:realm:password_hash" erstellen:
   htdigest [-c] passwdfile realm username
   Der Pfad zu "passwdfile" muss dann im .htaccess file angegeben werden

Settings:

- AuthType: Authentcation Type (digest/basic)
- AuthName: Realm name. Muss mit realm im password-file übereinstimmen.
- AuthDigestAlgorithm: Hash Algorithmus (MD5)
- AuthDigestNonceLifetime: Zeitraum für den der aktuelle Zufallswert gültig ist (in s)
- AuthDigestDomain: Domains für die diese Authentifizierung gültig ist
- AuthDigestQop: Quality of Protection. Nur username/passwort oder + integrity check
- AuthUserFile: Pfad zum password file
- Require: Gibt Voraussetzungen für Zutritt an: all granted, alldenied, valid-user(mit pwd file), user, group.

Aufgabe 3

Machen Sie sich mit dem Firefox Web Developer vertraut, insb. mit der 'Network' An- sicht. Öffnen Sie eine grössere Seite (z.B. www.uni-salzburg.at) welche HTTP Status Codes treten auf? Was bedeuten Sie?
Site: uni-salzburg.at
Statuscodes:

- 200 OK: Angefragter Inhalt wird ausgeliefert.
- 301 Moved Permanently: Permanenter Redirect. In diesem Fall eine Weiterleitung vom Port 80 (http) auf Port 443 (https).
  Auf dieser Seite wird man zusätzlich noch einmal weitergeleitet nämlich von https://uni-salzburg.at/ auf https://uni-salzburg.at/index.php?id=....
  Erst danach wird der content ausgeliefert.
- 302 Moved Temporarily: Redirect auf definierte Error-Seite bei request einer nicht vorhandenen datei.

- 404 Not Found: Tritt nie auf. Immer redirect auf eine vordefinierte Error Seite.

Aufgabe 4

Wie sind HTTP 1.1 request und response prinzipiell aufgebaut? Illustrieren Sie mit einem eigenen Beispiel!
Aufbau:

```
Request = Request-Line
          headers
          CRLF
          message body

Request-Line = Method SP Request-URI SP HTTP-Version CRLF
Method           = "PUT" | "GET" | "POST" | .....
RequestURI   = Pfad zur angefragten Datei
HTTP-Version = Protokoll version

headers          = Accept-* | Authorization | Expect | From | Host
                | User-Agent | Max-Forwards | cache-control |....

Response = Status-Line
           headers
           CRLF
           message-body

Status-Line   = HTTP-Version SP Status-Code SP Reason-Phrase CRLF
Status-Code   = "1xx", "2xx", "3xx", "4xx", "5xx"
Reason-Phrase = "OK", "Not Found", "Permanent Redirect"

headers = Accept-Ranges | Age | ETag | Location | Proxy-Authenticate
                | Retry-After | Server | Vary | WWW-Authenticate |....
```

Quelle: https://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html
Beispiele (apple.com):
Request:

```
Request URL:https://www.apple.com/
```

```
Request Method:GET
Accept:text/html,application/xhtml+xml,application/xml;q=0.9,
image/webp,image/apng,*/*;q=0.8
Accept-Encoding:gzip, deflate, br
Accept-Language:en-GB,en-US;q=0.9,en;q=0.8
Cache-Control:max-age=0
Connection:keep-alive
Cookie:key=value....
Host:www.apple.com
Upgrade-Insecure-Requests:1
User-Agent:Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/64.0.3282.186 Safari/537.36 OPR/51.0.2830.55
```

Response:

```
Status Code:200 OK
Cache-Control:max-age=0
Connection:keep-alive
Content-Encoding:gzip
Content-Length:6434
Content-Type:text/html; charset=UTF-8
Date:Wed, 14 Mar 2018 13:02:58 GMT
Expires:Wed, 14 Mar 2018 13:02:58 GMT
Server:Apache
Vary:Accept-Encoding
X-Content-Type-Options:nosniff
X-Frame-Options:SAMEORIGIN
X-Xss-Protection:1; mode=block
```

Aufgabe 5

Gruppe: Was bedeuten folgende Felder im HTTP request und response header? DNT, Connection: keep-alive, ETag, Content-Length

- DNT: Befiehlt einer Seite, den Nutzer nicht zu tracken.
- Connection: keep-alive: Mit dem Feld wird die Verbindung nicht sofort abgebrochen, sondern bis einer der beiden die Verbindung abbrechen will.

- ETag: (entity tag)Das Feld dient zur Bestimmung von Änderungen an der angeforderten Ressource und wird hauptsächlich zum Caching verwendet. Dies bedeutet dass bei der ersten Anfrage der ETag vom Server mitgeschickt wird und sollte nochmal eine Anfrage auf die selben Ressourcen geschickt werden wird der ETag vom Client geschickt und sollte es sich um zweimal den selben Wert handeln schickt der Server die Ressourcen nicht nochmal sondern weist darauf hin, dass die Ressourcen noch immer die selben sind.

- Content-Length: Gibt die Länge des Bodys in Bytes an.

Aufgabe 6

Gruppe: Wie unterscheidet sich HTTP 1.1 von HTTP/2? Wie wird ausgehandelt welches Protokoll gesprochen wird? Zeichnen Sie den Netzwerkverkehr auf Hinweis: tcpdump + Wireshark oder Web Developer

HTTP 1.1 baut normalerweise immer mehrere TCP-Verbindungen auf um jede Datei oder Bild einzeln zu übertragen. In HTTP/2 soll dies anders sein. Nach dem die erste TCP-Verbidnung aufgebaut wurde bleibt es bei der Verbindung und der Server erhält das Recht die Skripte und Style sheets unaufgefordert zu schicken. Dies reduziert vor allem das Head-of-Line Blocking, bei der der Browser normalerweise zu erst die Skripts und sheets durchscannt bevor er eine erneute Anfrage versenden kann. Bei HTTP/2 wird eine Verbindung aufgebaut durch die pausenlos Daten fließen.

Weitere Unterschiede:

- binär, nicht textuell: (kompakter)

- Multiplexed, parallelisiert: Mehrere Transfers über eine Verbindung

- Header Kompression: weniger Daten, kleinerer Overhead

- Server Push: Potentiell schnellere Response-Time. Inhalte können direkt von Server mit gepusht werden. HTTP/1.1: html vom browser geparsed und dann erst andere Inhalte requested.

Das Aushandeln der Verbindung funktioniert folgendermaßen:

1. Der Client sendet eine gewöhnliche HTTP/1.1 Anfrage mit "Upgrade"-Header:
   GET / HTTP/1.1

Host: server.example.com
Connection: Upgrade, HTTP2-Settings
Upgrade: h2c
HTTP2-Settings: ¡base64url encoding of HTTP/2 SETTINGS payload¿
"h2c" steht hierbei für unverschlüsseltes HTTP/2. Zusätzlich wird exakt ein HTTP/2 Header-Feld angegeben.

2. Ein Server, der HTTP/2 nicht versteht, führt keinen Upgrade durch und sendet eine gewöhnliche HTTP/1.1 Response:
   HTTP/1.1 200 OK
   Content-Length: 243
   Content-Type: text/html
   ...

3. Versteht der Server HTTP/2, wird eine "Protocol Switching"-Response zurückgegeben und ab diesem Zeitpunkt HTTP/2 verwendet:
   HTTP/1.1 101 Switching Protocols
   Connection: Upgrade
   Upgrade: h2c
   HTTP/2 connection...

Quelle: https://community.akamai.com/community/web-performance/blog/2016/06/22/underst
how-the-http2-protocol-is-negotiated

Leider konnte kein Server gefunden werden, der Requests auf HTTP/2 upgraded, deshalb wurde ein Beispiel-Capture heruntergeladen (siehe h2-14-plain-nghttp2.pcapng). In diesem ist nach Filterung auf "http" der Anfrageverlauf mit Upgrade auf HTTP/2 sichtbar.
Quelle: https://daniel.haxx.se/http2/h2-14-plain-nghttp2.pcapng