Anleitung zur Installation/Einrichtung eines Servers für die Veranstaltung 'Webprogrammierung'

26. März 2019

Inhaltsverzeichnis

Anmerkung

Diese Anleitung enthält sicherlich Fehler. Anmerkungen und Verbesserungsvorschläge daher gerne an:

- Stephan Wiefling (stephan.wiefling@th-koeln.de)
- Hoai Viet Nguyen (viet.nguyen@th-koeln.de)

Kapitel 1

Erneuerung des SSH Schlüsselpaars auf einem virtuellen Server mit Ubuntu Server

In unserem Szenario haben wir einen privaten Schlüssel für unseren virtuellen Server mit dem Betriebssystem Ubuntu Server als *.pem* Datei vorliegen. Dieser wird benutzt, um sich in diesen virtuellen Server über das Netzwerkprotokoll *Secure Shell* (SSH) einzuloggen. Nun soll das Schlüsselpaar zum Login via SSH geändert werden. Im Folgenden wird beispielhaft beschrieben, wie dabei vorzugehen ist. Der private Schlüssel für den Praktikumsserver steht in ILIAS als Download zur Verfügung.

Zunächst benötigen wir ein Softwareprogramm, um uns auf den virtuellen Server einzuloggen (SSH Client). Auf Mac OS X und Linux-Betriebssystemen sollte ein SSH Client bereits standardmäßig installiert sein. Für Windows bietet sich PuTTY an, welchen wir für diese Anleitung verwenden werden. Er kann kostenfrei über diese URL bezogen werden (*PuTTY* und *PuTTYgen* müssen dabei heruntergeladen werden): `http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html`

Zur Dateiübertragung zwischen unserem Rechner und dem virtuellen Server, benötigen wir einen *SFTP Client*. In dieser Anleitung verwenden wir *Cyberduck*, welcher für Windows und Mac OS X unter `https://cyberduck.io` verfügbar ist.

Für Linux Nutzer bietet sich Filezilla an. Auf Debian basierten Linux-Systemen wie z.B. Ubuntu lässt er sich in der Kommandozeile über

```
sudo apt-get install filezilla
```

installieren. Wir werden im Folgenden nur auf die Vorgehensweise mit Cyberduck eingehen. Es sei aber darauf hingewiesen, dass die Vorgehensweise in Filezilla sehr ähnlich ist.

1.1 Login auf dem Server

Nach der Installation der SSH Clients (auf Windows) und des SFTP Clients, werden wir uns zunächst auf dem virtuellen Server einloggen. Dafür erforderlich ist:

- Die *IP Adresse*, unter der der Server erreichbar ist
- Der *Private Key* zur Authentifizierung mit dem Server

1.1.1 Mac OS X / Linux

Zunächst öffnen wir das Terminal und navigieren zu dem Ort, an dem die bereitgestellte Private Key Datei gespeichert wurde. Liege der Private Key mit dem Dateinamen *ubuntu.pem* beispielsweise im Ordner *Downloads* des aktuell auf dem Rechner eingeloggten Benutzers, erreichen wir den Ordner mit:

```
cd Downloads
```

Anschließend müssen wir die Zugriffsrechte für die Private Key Datei so beschränken, dass nur der Besitzer der Datei Zugriff darauf hat. Dies erreichen wir in unserem Beispiel mit

```
chmod 600 ubuntu.pem
```

Ob die Zugriffsrechte geändert wurden, überprüfen wir mit Eingabe von:

```
ls -l
```

In der Ausgabe sollten wir nun eine ähnliche Ausgabe wie diese hier sehen:

```
-rw-------@ 1 user  staff        1679 21 Apr 09:39 ubuntu.pem
```

Der Wert `rw` in der Linken Spalte, gefolgt von ausschließlich $--$ zeigt an, dass nur unser Benutzer Lese- und Schreibrechte für die Datei besitzt.

Nun können wir uns in den virtuellen Server einloggen. Der Benutzername für den virtuellen Server ist in unserem Fall standardmäßig `ubuntu`. Zusammen mit der Schlüsseldatei und der Server IP Adresse loggen wir uns im Beispiel nun in den Server über SSH ein:

```
ssh -i ubuntu.pem ubuntu@[Server IP Adresse]
```

Die daraufhin folgende Sicherheitsabfrage bestätigen wir mit `yes`. Anschließend wird eine Verbindung zum Server hergestellt. Dies kann gegebenenfalls etwas dauern. Sobald wir eingeloggt sind, sollte das Terminal in etwa so aussehen:

```
ubuntu@WebPraktikum:~$
```

1.1.2 Windows

In Windows müssen wir zunächst die Private Key Datei mit der Endung *.pem* in eine *PuTTY Key Datei* mit der Endung *.ppk* umwandeln. Hierzu öffnen wir zunächst PuTTYgen und danach im Programmfenster in der Leiste *Conversions* $\rightarrow$ *Import Key*. Im dann erscheinenden Fenster wählen wir die .pem Datei aus. Sie wird dann in PuTTYgen importiert. Anschließend klicken wir auf *Save private key* und bestätigen die anschließende Sicherheitsfrage mit Ja. Danach wählen wir den Speicherort für die Datei aus und haben danach unser .ppk File generiert.

Nun öffnen wir PuTTY und bekommen dann das Fenster wie in Abbildung 1.1 zu sehen. Unter *Host Name* tragen wir dann die IP Adresse unseres virtuellen Servers ein. Anschließend wählen wir auf der

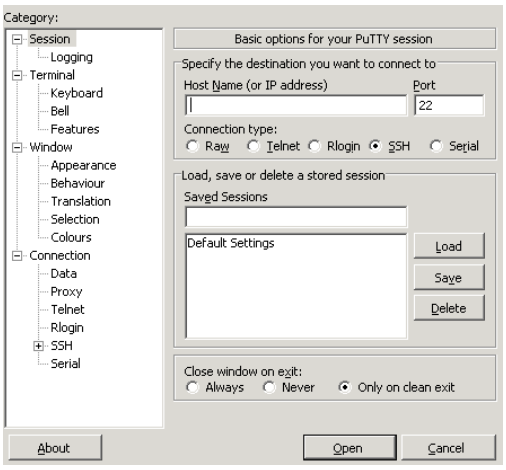

Abbildung 1.1: Einstellungen Putty: Session

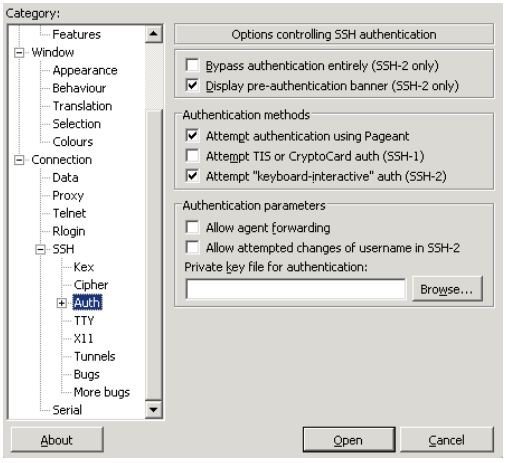

Abbildung 1.2: Einstellungen Putty: Auth

Linken Seite den Reiter *Connection* → *SSH* → *Auth* aus. Wir bekommen nun das Fenster aus Abbildung 1.2 zu sehen. Unter *Private key file for authentication* wählen wir die vorher erzeugte .ppk Datei aus. Anschließend klicken wir auf *Open*.

Die Sicherheitsfrage bestätigen wir mit *Ja* und tippen im Terminalfenster auf die Frage *login as:* `ubuntu` ein und bestätgen mit Enter. Wenn der Login erfolgreich war, sollte der Fensterinhalt in etwa diese Zeilen beinhalten:

```
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.13.0-85-generic x86_64)
...
ubuntu@WebPraktikum:~$
```

1.2 Einloggen über SFTP

Sobald wir uns über SSH eingeloggt haben, werden wir uns zusätzlich über SFTP auf dem virtuellen Server einloggen. Die Schritte für Mac OS X und Windows in Cyberduck sind dabei identisch.

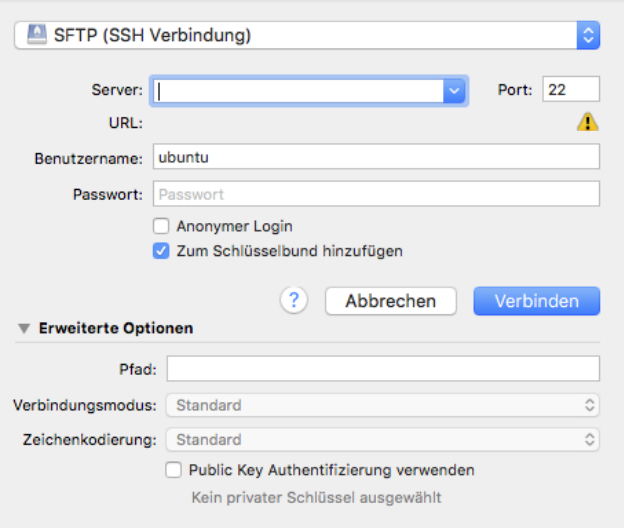

Abbildung 1.3: Einstellungen Cyberduck

Zunächst klicken wir im geöffneten Cyberduck oben Links auf *Neue Verbindung*. Im daraufhin erscheinenden Fenster füllen wir die Werte wie in Abbildung 1.3 aus. Bei Server tippen wir die Server IP Adresse unseres virtuellen Servers ein. Danach machen wir unten einen Haken bei *Public Key Authentifizierung verwenden*. Dabei sollte ein Dateifenster erscheinen, in welchem wir die Private Key Datei (*.pem*, nicht *.ppk*) auswählen. Anschließend klicken wir wieder im Einstellungsfenster auf *verbinden*. Cyberduck stellt nun einer Verbindung zum Server her. Ist sie erfolgreich, sieht das Fenster ähnlich wie in Abbildung 1.4 aus.

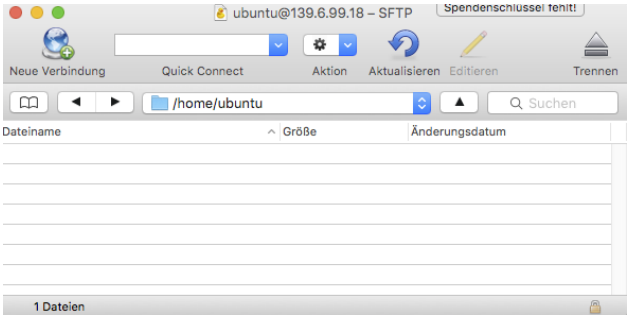

Abbildung 1.4: Cyberduck nach erfolgreicher SFTP-Verbindung

1.3 Generierung des neuen Schlüsselpaars

Nach erfolgtem SFTP Login können wir einen neuen privaten Schlüssel für den SSH Zugriff auf dem virtuellen Server erzeugen. Dazu wechseln wir wieder zu unserem Terminalfenster mit der aktuellen SSH

Verbindung.

Nun erzeugen wir ein neues SSH Schlüsselpaar mit dem asymmetrischen Verschlüsselungsverfahren RSA und einer Schlüssellänge von 4096 Bits.

```
ssh-keygen -t rsa -b 4096
```

Der Server fragt dabei nach einem Speicherort für die Schlüsseldateien, welchen wir so belassen können und daher mit Enter bestätigen. Die Eingabe eines Passworts ist nicht zwingend erforderlich und kann daher auch mit Enter bestätigt werden. Danach ist das Schlüsselpaar mit Public und Private Key erzeugt. Die Speicherorte werden ebenfalls ausgegeben:

```
Your identification has been saved in /home/ubuntu/.ssh/id_rsa.
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub.
```

Nach der Erzeugung müssen wir unseren Public Key in die SSH-Zugriffsliste des virtuellen Servers eintragen. Dies machen wir über den Befehl

```
cat /home/ubuntu/.ssh/id_rsa.pub >> /home/ubuntu/.ssh/authorized_keys
```

welcher den Inhalt des öffentlichen Schlüssels *id_rsa.pub* in die Zugriffsliste *authorized_keys* einträgt.

Als Folge haben wir nun zwei Zugriffsschlüssel für unseren virtuellen Server in der Liste: Den Private Key vom Anfang der Anleitung und den gerade erzeugten Schlüssel. Letzteren müssen wir nun auch auf unseren eigenen Rechner kopieren, um ihn benutzen zu können. Hierzu verschieben wir das erzeugten Private Key File *id_rsa* (Achtung: Ohne ".pub" am Ende) in den Home-Ordner.

```
mv /home/ubuntu/.ssh/id_rsa /home/ubuntu/
```

Dann wechseln wir wieder zum geöffneten Cyberduck-Fenster und drücken auf *Aktualisieren*. Der Private Key sollte nun sichtbar sein, wie in Abbildung 1.5 zu sehen. Diesen laden wir auf unseren Rechner und löschen ihn anschließend vom Server mit Cyberduck. Der Übersicht halber sei empfohlen, dem Private Key einen eindeutigen Namen zu geben und ihn z.B. auf dem eigenen Rechner in *WebPraktikum.pem* umzubenennen.

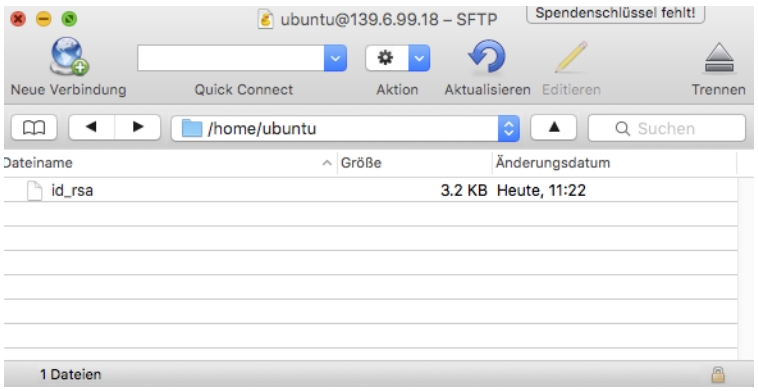

Abbildung 1.5: Cyberduck mit Private Key

Anschließend wechseln wir wieder ins Terminal und starten den SSH-Server neu.

```
sudo service ssh restart
```

Jetzt können wir die Verbindung zum Server schließen mit:

```
exit
```

Ebenso trennen wir die SFTP-Verbindung in Cyberduck.

1.4 Login mit neuem Schlüsselpaar

Nun wollen wir überprüfen, ob wir uns mit dem neu generierten Private Key im virtuellen Server einloggen können. Dazu wiederholen wir in Windows die selben Schritte in PuTTYgen (Schlüsselumwandlung) und PuTTY, nur verwenden wir diesmal den neu erstellten Private Key zur Authentifizierung.

In Mac OS X und Linux hingegen müssen wir noch einen weiteren Schritt vorher tätigen und den Server aus der Datei */.ssh/known_hosts* im Home-Verzeichnis löschen. Dazu löschen wir einfach die letzte Zeile dieser Datei mit:

```
sed '$d' ~/.ssh/known_hosts > ~/.ssh/known_hosts
```

Anschließend wiederholen wir die Schritte für den SSH Login, wie in Abschnitt 1.1.1 beschrieben, allerdings verwenden wir die neue Schlüsseldatei zur Authentifizierung. In unserem Beispiel also:

```
ssh -i WebPraktikum.pem ubuntu@[Server IP]
```

1.5 Löschen des alten Public Keys

Können wir uns mit dem neuen Private Key in den Server einloggen, dann ist die neue Einstellung korrekt. Damit der alte Private Key nicht mehr zur Authentifizierung verwendet werden kann, müssen wir den Eintrag des alten Public Keys in *.ssh/authorized_keys* im Home-Verzeichnis des Servers entfernen. Dazu öffnen wir auf dem virtuellen Server mit aktiver SSH-Verbindung den Editor *nano* im Terminal mit der entsprechenden Datei:

```
nano ~/.ssh/authorized_keys
```

Im nun erscheinenden Editor-Fenster löschen wir den Inhalt der ersten Zeile. Der ursprüngliche Inhalt der zweiten Spalte sollte sich nun in der ersten Spalte befinden. Jetzt schließen wir den Editor mit der Tastenkombination *Strg* + *X*, bestätigen die Speicherfrage mit *Y* und drücken *Enter*. Anschließend starten wir den SSH Server wieder neu.

```
sudo service ssh restart
```

Ein Login mit dem alten Private Key sollte nun nicht mehr möglich sein. Zur Bestätigung können wir dies mit einem Loginversuch mit dem alten Private Key testen.

Für Mac OS X und Linux könnt es z.B. so aussehen:

```
ssh -i ubuntu.pem ubuntu@[Server IP]
Permission denied (publickey).
```

Anmerkung

Möchte man seinen Server im besonderem Maße vor unbefugtem Zugriff schützen, ist die beschriebene Methode nicht zu empfehlen, da wir den Private Key über eine Internetleitung transferieren. Für den produktiven Einsatz ist es empfehlenswert, den Private Key auf einem separatem Rechner zu erzeugen und nur den Public Key auf den Server zu übertragen.

Kapitel 2

Installation von Apache HTTP Server auf Ubuntu

Im vorherigen Kapitel haben Sie sich bereits mit der Erneuerung bzw. der Generierung von Schlüsselpaaren und dem entfernten Einloggen auf einen Ubuntu Server vertraut gemacht. In diesem Kapitel wollen wir Ihnen zeigen wie Sie einen Apache HTTP Server mit dem Paketmanager von Ubuntu installieren können.

2.1 Linux Paketmanager

Die meisten Linux-Distributionen besitzen einen Paket-Management-System mit dem man Software in Form von Paketen bequem heruntergeladen, installieren, updaten und deinstallieren kann. Bekannte Beispiele wären da der Red Hat Package Manager (RPM) zu nennen, der in vielen GNU/Linux-Distributionen zum Einsatz kommt. Debian-basierte Betriebssysteme und somit auch Ubuntu bringen den Debian Package Manager (dpkg) vorinstalliert mit.

2.1.1 Advanced Packaging Tool auf Ubuntu

Das Advanced Packaging Tool oder auch kurz APT ist ebenfalls eine Paketverwaltungssystem das mit dpkg arbeiten und auch die Verwaltung von RPM-Paketen erlaubt. Ebenso ist dieses Tool bereits bei Ubuntu und Debian vorhanden und besteht aus mehreren Programmen. Im Praktikum werden wir hauptsächlich *apt-get* für die Paketverwaltung und *apt-cache*, das uns sowohl Informationen über Pakete mitteilt als auch uns die Suche nach Pakten bzw. Programmen ermöglicht. Daneben existieren noch weiter APT-Kommandos wie z.B. *apt-key* zur Schlüsselverwaltung oder *apt-pinning*, welche für die Rangordnung der Paketequellen zuständig ist.

2.1.2 apt-cache

Der Befehl *apt-cache* erlaubt uns Metadaten von Paketen und die Paketdatenbank (Paketcache) anzuzeigen und zu verändern. Um nach bestimmten Paketen zu suchen benötigen wir das Kommando *apt-cache search*. Beispielweise suche wir hier im Paketcache nach dem Begriff "apache":

```
apt-cache search apache
```

Danach listet uns das Programm alle Pakete auf, die im Paketnamen oder in der Beschreibung das Wort "apache" enthalten. Es können aber auch mehrere Begriffe eingeben werden wie z.B:

```
apt-cache search apache proxy
```

Hier zeigt uns APT alle vorhanden Pakete an, die die Begriffe "apache" und "proxy" aufweisen.

apt-cache show gibt uns genauere Informationen über das Paket her.

```
apt-cache show apache2
```

Dieser Befehl ermittelt uns eine detailliertere Beschreibung über das jeweilige Paket "apache2".

Weiter Informationen über *apt-cache* erfahren Sie hier: `http://wiki.ubuntuusers.de/apt/apt-cache`

2.1.3 apt-get

Wenn Sie nun Pakete installieren, aktualisieren oder löschen möchten, steht Ihnen der Befehl *apt-get* zu Verfügung. Pakete lassen sich mit dem Kommando *apt-get install* herunterladen und installieren. Also z.B.:

```
apt-get install apache2
```

würde demnach das die Version 2 des Webservers Apache installieren. Auch hier können mehrere Pakete hintereinander eingegeben werden. Beachten Sie bitte, dass viele Pakete auf Ubuntu-System super-user-Rechte erfordern. Dies ist den Administratorrechten von Microsoft Windows ähnlich. Geben Sie dazu bitte vor jedem entsprechenden Befehl *sudo* mit an oder wechseln direkt den Benutzer mit *sudo su*. Z.B:

```
sudo apt-get install apache2
```

oder:

```
sudo su
apt-get install apache2
```

Mit dem Befehl *apt-get remove* deinstallieren Sie Pakete. Auch hier ist die Angabe von mehreren Paketen möglich. Für das Praktikum benötigen wir noch den Befehl *apt-get update*, der die Liste der eingetragenen Paketquellenlisten aktualisiert. Dieser Befehl muss ebenfalls mit super-user-Rechten ausgeführt werden, erfordert aber keine Angabe von einen Paketnamen. Wichtige Kommandos wären noch *apt-get upgrade* und *apt-get dist-upgrade*, die für die Aktualisierung der Pakete wichtig sind. Auch diese beiden Befehl erfordern super-user-Rechte. Nähere Beschreibungen über *apt-get* gibt es hier: `http://wiki.ubuntuusers.de/apt/apt-get`

2.2 Apache2 HTTP Server auf Ubuntu

Da wir uns bereits über die Werkzeuge zur Installation von Programmen in Linux-System beschäftigt haben, können uns jetzt um die wichtigen Komponenten, die in der Webentwicklung wesentlich sind, widmen. Jede Webseite, jede Webanwendung oder jedes web-basierte, verteilte System benötigt einen Webserver. Der meist genutzte Webserver ist der Apache Webserver von der Apache Software Foundation, der sich derzeit in der Version 2 befindet. Dieser Webserver erlaubt es in erste Linie nur statische Dateien auszutauschen. Um dynamische Webseiten zu generieren mittels z.B. PHP oder Python, müssen noch zusätzliche Module installiert oder aktiviert werden.

2.2.1 Apache2 Installation

Bevor Sie die neueste Version von Apache installieren wollen, müssen Sie die Paketquellenliste auf den neusten Stand bringen. Dies erfolgt durch den Befehl:

```
sudo apt-get update
```

Danach können Sie mit:

```
sudo apt-get install apache2
```

die aktuelle Version von Apache Webserver installieren. Wenn Sie nun die IP Adresse Ihres Servers in die Adressleiste Ihres Browsers eingeben, sollten Sie den Apache Webserver ansprechen können (siehe Abbildung 2.1).

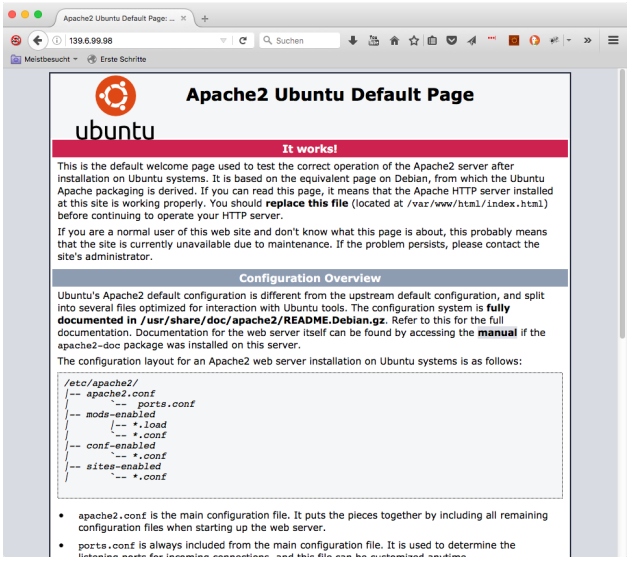

Abbildung 2.1: Aufruf einer statischen Webseite auf den Apache Webserver

2.2.2 Apache2 Ordnerstruktur

Die Installation von Apache legt Ihnen einen Ordner **apache** in das Verzeichnis **/etc** an. Wenn wir in diesen Ordner mit dem Befehl:

```
cd /etc/apache2
```

wechseln und uns dort die Ordnerstruktur mit *ls* anschauen sehen wir Folgendes:

```
apache2.conf    conf-enabled  magic           mods-enabled  sites-available
conf-available  envvars       mods-available  ports.conf    sites-enabled
```

Abbildung 2.2: Ordnerstruktur von Apache

Die Datei **apache2.conf** beinhaltet die hauptsächlichen und globalen Konfiguration. Daneben existieren noch der **conf-available**-Ordner, wo sich zusätzliche Konfigurationsdateien befinden und der **conf-enabled**-Ordner, welches die aktiven Konfigurationfiles umfasst. Bei den Dateien im **conf-enabled**-Ordner handelt es sich um Symbolische Verknüpfungen der Dateien aus dem **conf-available**-Verzeichnis. Gleiches spiegelt sich auch in den **mods-available** und **mods-enabled** wieder, wo vorhandene Module und aktive Module aufgelistet sind. Die Ordner **site-available** und **site-enabled** zeigen ebenfalls verfügbare Virtualhost und aktivierte Virtualhost an. Die Datei **ports.conf** spezifiziert die anzusprechenden Ports des Webservers. So kann man z.B. zusätzlich zum Standardport 80 noch weitere Ports hinzu schalten wie z.B. den Port 443 für TLS/SSL-Verbindungen. Die **magic**-Datei hilft Apache dabei MIME-Types aus dem Inhalt heraus zu bestimmen. In **envvars** sind Umgebungsvariablen festgelegt, welches für das mitinstallierte Steuerungs-Interface *apache2ctl* wichtig ist.

2.2.3 Apache2 Konfiguration

Vor jeder Inbetriebnahme einer Webseite oder Webanwendung müssen wichtige Konfiguration festgelegt und geändert werden. Hier ist es essentiell kritische Information zu verstecken. Der Apache Webserver sendet im jedem Antwort-Header Details über die Server mit. Entsprechende Entwickler-Tools von Firefox und Chrome (siehe Abbildung 2.3), erlauben es diesen Header auszulesen.

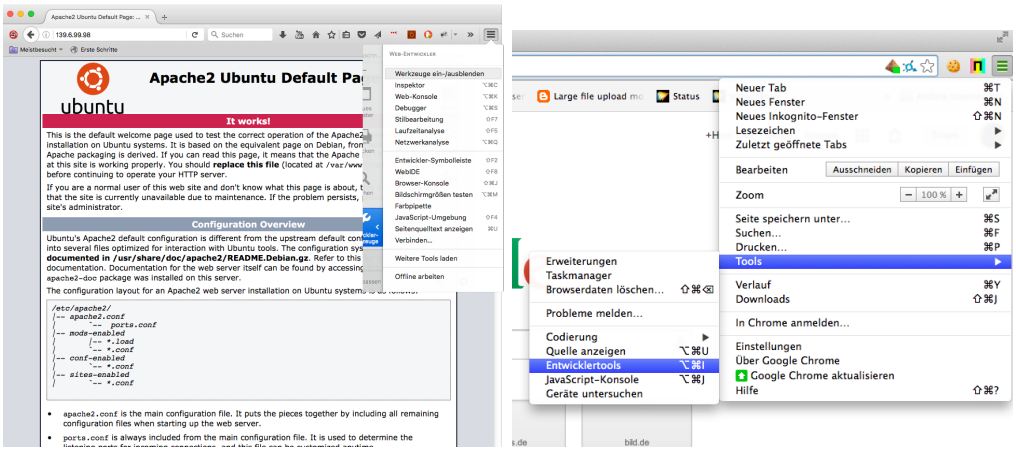

Abbildung 2.3: Firefox/Chrome Developer Tools

Der Header beinhaltet den Namen des Server, die Versionsnummer und das Betriebssystem (siehe Abbildung 2.4). Dieser kleine aber wichtige Information sollten Betreiber unbedingt ausblendet, da sonst

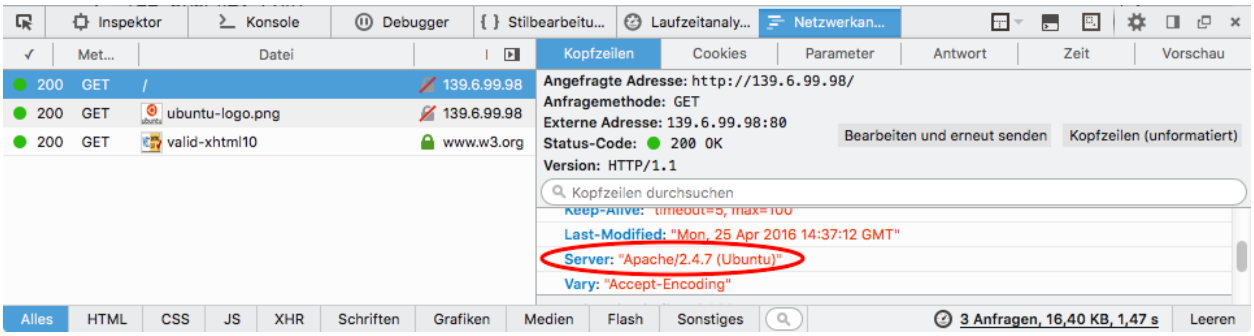

Abbildung 2.4: Server-Information im Header

Angreifer über die Versionsnummer nach gezielt nach Sicherheitslücken suchen können. Um diesen Header zu ändern wechseln Sie dazu bitte in den **conf-enabled**-Ordner und bearbeiten Sie die Datei **security.conf**. Als Texteditor eignet sich z.B. *nano* oder *vi*. *nano* ist einfacher zu benutzen, dafür hat *vi* eine steile Lernkurve, besitzt aber dafür mächtigere Werkzeuge. Für das Praktikum reicht aber *nano* völlig aus. Öffnen Sie nun mit super-user-Rechten die entsprechende Datei:

```
cd conf-enabled
sudo nano security.conf
```

Die geöffnete Datei beinhaltet Sicherheitskonfiguration des Webservers. Alle Zeilen die mit "#" beginnen, repräsentieren Kommentare und werden vom Apache nicht interpretiert. Suchen Sie bitte nach dem Begriff "ServerTokens OS". Ändern Sie die Zeile in "ServerTokens Prod" um und speichern Sie die mit den Tasten STEUERUNG+O und ENTER ab. Anschließend schließen Sie bitte den Editor mit STEUERUNG+X. Der Apache Webserver erfordert nun einen Neustart, um die veränderten Konfiguration zu übernehmen. Dies erfolgt durch:

```
sudo apache2ctl restart
```

Vergewissern Sie sich nochmal über diese Änderung übernommen wurde, indem Sie die Seite im Browser neuladen mit den entsprechenden Entwickler-Tools den Header auslesen (siehe Abbildung 2.5).

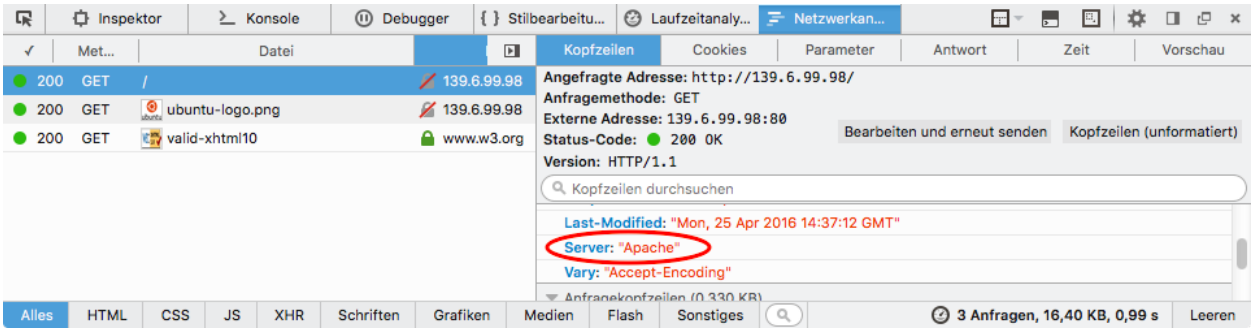

Abbildung 2.5: Header zeigt nur den Servernamen an

Es können noch weitere Sicherheitseigenschaften in eingestellt werden, die **security.conf**-Datei kommen-

tiert diese entsprechenden Einstellungen. Weiter wesentliche Information in den Konfigurationsdateien finden Sie in **sites-enabled**-Ordner. Dort befindet sich die Datei **000-default.conf**, die einen Virtualhost definiert. Beachten Sie bitte, dass der Name der Datei bei Ihren Betriebssystem abweichen kann. Wenn Sie diese Datei öffnen finden ähnliche Informationen wie in Listing 2.1.

```
<VirtualHost *:80>
# The ServerName directive sets the request scheme, hostname and port that
# the server uses to identify itself. This is used when creating
# redirection URLs. In the context of virtual hosts, the ServerName
# specifies what hostname must appear in the request's Host: header to
# match this virtual host. For the default virtual host (this file) this
# value is not decisive as it is used as a last resort host regardless.
# However, you must set it for any further virtual host explicitly.
#ServerName www.example.com

ServerAdmin webmaster@localhost
DocumentRoot /var/www/html

# Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
# error, crit, alert, emerg.
# It is also possible to configure the loglevel for particular
# modules, e.g.
#LogLevel info ssl:warn

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

# For most configuration files from conf-available/, which are
# enabled or disabled at a global level, it is possible to
# include a line for only one particular virtual host. For example the
# following line enables the CGI configuration for this host only
# after it has been globally disabled with "a2disconf".
#Include conf-available/serve-cgi-bin.conf
</VirtualHost>
```

Listing 2.1: Virtualhost Konfiguration

In dieser Datei müssen wir nichts ändern, dennoch weißt diese Datei darauf hin, dass sich die Document Root unseres Webservers im Ordner **/var/www/html** befindet. Dort können wir Dateien hinzufügen, ändern oder löschen und im Webbrowser abrufen.

2.2.4 Statische Dateien ändern und hinzufügen

Zum Ändern von der Startseite unseres Webservers wechseln Sie dazu bitte in das Verzeichnis **/var/www/html** und öffnen Sie bitte mit einem Texteditor Ihrer Wahl die **index.html**-Seite.

```
cd /var/www/html
sudo nano index.html
```

Hier kann die Startseite nach Ihrer Wahl geändert werden. Ändern Sie bitte diese Datei, indem Sie z.B. " Hello World " im "<Body>"-Tag hinzufügen. Speichern Sie diese Datei ab und rufen Sie die Seite erneut auf. Der Webserver muss hierbei nicht neugestartet werden.
Um neue statische Dateien zu erzeugen rufen Sie einfach den entsprechenden Editor mit dem Dateinamen auf.

```
sudo nano newFile.html
```

Fügen Sie einen entsprechenden Text ein und speichern Sie diese Datei ab. Anschließend können Sie diese Datei über den entsprechenden Pfad aufrufen:

`http://<IhreIP>/newFile.html.`

Natürlich ist es auch möglich, Dateien von Ihrem Rechner per FTP Hochzuladen. Dazu eignet sich der FTP-Client "Cyberduck". Nach Start des Programms klicken Sie bitte auf "Neue Verbindung". Dort öffnet sich eine Eingabemaske. Wählen Sie bitte ganz oben den Protokolltyp SFTP an, der Ihnen eine gesicherte Verbindung garantiert. Als Servernamen tragen Sie Ihre IP Adresse ein und der Benutzername ist ubuntu. Das Passwortfeld lassen wir leer (siehe Abbildung 2.6).

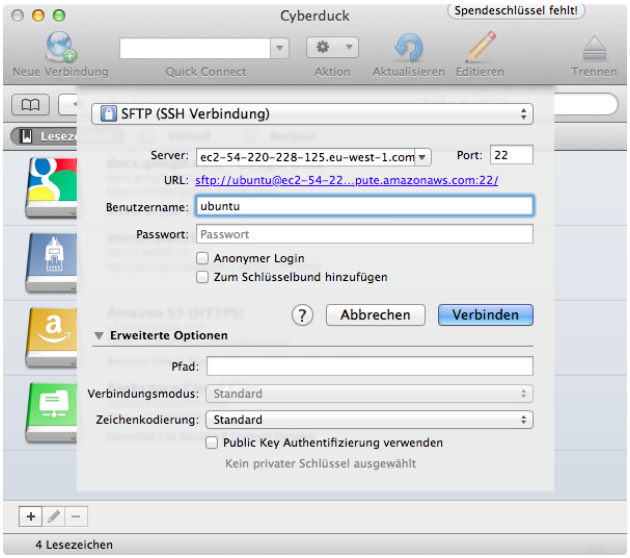

Abbildung 2.6: Cyberduck

Achten Sie bitte darauf, dass die Portnummer 22 eingetragen und das Häkchen "Anonymer Login" weggeklickt ist. Anschließend müssen wir unseren private Key noch hinzufügen. Diese Optionen findet sich in den "erweiteren Optionen", in denen Sie auf "Public Key Authentifizierung verwenden" klicken. Jetzt öffnet sich ein File-Explorer, in welchem Sie den Key anwählen können. Verwenden Sie dazu ausschließlich die ".pem"-Datei. Die Verbindung zum Server erfolgt durch den Klick auf "Verbinden". Hier öffnet sich das Home-Verzeichnis des Benutzers Ubuntu. Dort können Sie Dateien uploaden, indem Sie unter *Aktion->Upload* eine entsprechende Datei auswählen. Wichtig ist hierbei, dass sie nur in das Home-Verzeichnis des Benutzers Ubuntu Dateien hochladen können. Uploads in andere Ordner, darunter auch **/var/www/html**, sind ausschließlich mit Superuser-Rechten möglich. Diese Rechte können wir über das Terminal, jedoch nicht über "Cyberduck" erreichen. Daher laden wir zunächst mit "Cyberduck" die entsprechenden Dateien in einen Ordner im Home-Verzeichnis und verschieben diese anschließend im Terminal mit Superuser-Rechten in das Verzeichnis **/var/www/html**.

Das Home-Verzeichnis erreichen Sie im Terminal über den Befehl

```
cd ~
```

oder einfach

```
cd
```

Mit dem Kommando *ls* können Sie z.B. alle Dateien in diesem Ordner auflisten. Um die hochgeladen Dateien in die Document Root zu transferieren, benötigen Sie *mv* mit Superuser-Rechten:

```
sudo mv uploadedFile /var/www/html
```

Nachdem wir das durchgeführt haben, ist auch diese Datei im Webbrowser abgreifbar.

Kapitel 3

Installation von Go auf Ubuntu

Im Webpraktikum werden Sie eine Webanwendung in Go umsetzen. Die Installationsschritte sind in diesem Kapitel beschrieben.

3.1 Go Installation

Bevor Sie die neueste Version von Go installieren wollen, müssen Sie die Paketquellenliste auf den neusten Stand bringen. Dies erfolgt durch den Befehl:

```
sudo apt-get update
```

Danach können Sie mit:

```
sudo apt-get install golang-go
```

die aktuelle Version von Go installieren.

Erstellen Sie nun einen Arbeitsordner für die Go-Entwicklungsumgebung und fügen Sie diesen der Umgebungsvariable Ihres Server hinzu, in dem Sie folgende Befehle ausführen:

```
mkdir ~/go
echo "export GOPATH=$HOME/go" >> .bash_profile
echo "export GOBIN=$HOME/go/bin" >> .bash_profile
```

Starten Sie jetzt die Konsole neu, damit die Umgebungsvariable aktualisiert wird. Dazu müssen Sie diesen Befehl eingeben:

```
bash --login
```

Wenn Sie go in im Terminal ausführen sollte nun Folgendes erscheinen:

```
Go is a tool for managing Go source code.

Usage:
```

```
go command [arguments]

The commands are:

build       compile packages and dependencies
clean       remove object files and cached files
doc         show documentation for package or symbol
env         print Go environment information
bug         start a bug report
fix         update packages to use new APIs
...
```

Damit wurde Go installiert.