Teil der Diplomarbeit 'Open Source Webshop - Phpeppershop'

Studenten: José Fontanil und Reto Glanzmann

Dozent: Patrick Feisthammel

Erfahrungsbericht:

LAMPS einrichten

Bericht

Dies ist die komplett überarbeitete und Fehler bereinigte Version v.1.2 dieser Installationsanleitung.

Dieser Bericht ist ein Auszug aus der kompletten Diplomarbeitsdokumentation. Sie enthält viele Erfahrungen die wir gerne weitergeben möchten.

Inhaltsverzeichnis

Inhaltsverzeichnis

Installation der Systemumgebung - LAMPS....3
Inhalt dieses Kapitels....3
Wieso diese Kombination?....3
Installation....4
Quellen beschaffen....4
Linux....4
Apache (v.1.3.x)....4
MySQL....4
PHP....4
SSL....4
Erweiterungen....4
Eigentliche Installation....4
Entpacken der Pakete....4
MySQL....5
SSL Installation....5
Apache und mod_SSL....5
Server-Zertifikat erstellen....6
Apache Installation vervollständigen....7
PHP....7
PDFlib....7
Zlib....7
PHP konfigurieren und kompilieren....7
Konfiguration....8
Starten....8

Installation der Systemumgebung - LAMPS

Inhalt dieses Kapitels

Da wir zur Zeit unserer Diplomarbeit mit der aktuellen Version von PHP arbeiten wollten, haben wir uns entschieden, den Apache Server, PHP und SSL-Unterstützung neu zu installieren. Damit unsere Erfahrungen, die wir hier machten, auch an andere Leute weitergegeben werden können, haben wir diesen Erfahrungsbericht im Stil einer umfassenden Anleitung geschrieben.

Wieso diese Kombination?

Alle fünf Teile eines LAMPS-Systems: Linux, Apache, MySQL, PHP, (open-, mod_)SSL sind Projekte, welche gemeinsam haben, dass sie Open Source sind. Sie kosten nichts, sind erprobt und sehr leistungsfähig. Alle Projekte werden aktiv weiterentwickelt. Mit Linux als kostenlosem Betriebssystem, erhält man heute über eine Distribution (SuSE, Redhat, Debian, ...) ein zu kommerziellen Serverbetriebssystemen konkurrenzfähiges Produkt.

Apache ist ohnehin der am weitesten verbreitete Webserver der Welt (siehe Netcraft Analyse). Er lässt hinsichtlich Stabilität und Erweiterbarkeit keinerlei Wünsche offen. Insbesondere die Kombination von Apache mit PHP ist heutzutage sehr populär und auch bei den meisten Providern anzutreffen. ...Ausserdem ist er nicht anfällig auf Code Red...

MySQL ist eine kleine, aber im Web-Umfeld weit verbreitete Datenbank. Sie bietet zwar keine referentielle Integrität (MyISAM) und auch sonst kann sie nicht unbedingt mit einer IBM DB2 oder einer Oracle Datenbank konkurrieren. Ihr grosser Vorteil ist aber die grosse Unterstützung in der Web Open Source Gemeinde und ihre Geschwindigkeit. Es gibt unzählige Projekte, welche MySQL als Datenbank nutzen. Dies liegt nicht zuletzt auch daran, dass MySQL sehr einfach zu handhaben ist. Hier seien in diesem Zusammenhang noch die Projekt phpMyAdmin (http://www.phpwizard.net/phpMyAdmin/) und DB-Designer (http://fabforce.net) erwähnt, welche eine sehr komfortable Oberfläche zur Datenbank administration bietet.

PHP ist eine leistungsfähige Skriptsprache, primär dazu ausgelegt im Web-Umfeld verwendet zu werden. PHP geniesst in seiner vierten Inkarnation eine breite Unterstützung, einerseits durch Webserver-Einbindungen, andererseits aber auch durch zahlreiche Funktionen, die inzwischen in PHP implementiert wurden. PHP ist einfach zu erlernen, benutzt die vielfach verwendete ANSI-C Syntax und unterstützt von Haus aus viele Datenbanken. Die für das Webscripting unumgänglichen String-Operationen hat es zum grössten Teil von Perl übernommen. Die Einbindung des PHP Codes in HTML erfolgt analog zu Microsofts ASP. Auf diese Weise wird von allen bekannten Sprachen und Techniken 'das Beste' verwendet und in PHP vereint. PHP 5 bietet weiterhin ausgereifte OO-Eigenschaften.

Wenn man mit Geld hantieren will, kommt es auf Sicherheit an, und die Anforderungen an sie sollte man nicht unterschätzen. Die Datenübertragung von Kreditkartennummern, und Benutzerdaten und ihren Einkäufen dürfen für Dritte nicht einsehbar sein. Das heisst, dass man primär die Datenübertragung über das 'unsichere' Internet schützen, sprich verschlüsseln, muss. Eine heute anerkannte Technologie um dies zu realisiern ist SSL (Secure Socket Layer), neu TLS genannt. OpenSSL ist eine Open Source Implementation der Verschlüsselungsfunktionalität, welche vom mod_SSL (Apache DSO-Modul) benutzt wird.

Anbei noch bemerkt: Es wird hier nicht der *Apache 2* verwendet, weil PHP zum Zeitpunkt der Erstellung dieses Dokuments noch nicht aus dem Experimental Stadium heraus war.

Installation

Quellen beschaffen

Die beschriebenen Quellen beziehen sich auf Pakete, welche im August 2003 aktuell waren. Natürlich werden die Pakete zu einem späteren Zeitpunkt andere Versionsnummern tragen.

Linux

Linux beschafft man sich am besten als Komplettpaket in Form einer Distribution. Wir arbeiteten mit *SuSE Linux Professional 8.2*. Für welche man sich entscheidet, ist grundsätzlich Geschmackssache und im speziellen für diese Anwendung eigentlich nicht von Bedeutung. Wir benutzen hier weder das X-System, noch distributionsspezifische Eigenheiten. Vorausgesetzt wird lediglich ein lauffähiges Linux-System (natürlich mit gcc, flex, bison, u.s.w.).

Wir empfehlen die folgenden Pakete gleich ins Verzeichnis `/usr/local/src/` zu speichern, dann können wir vom gleichen Pfad ausgehen. Ausserdem ist so durch die Verzeichnisstruktur festgelegt, dass es sich um Sourcepakete handelt.

Apache (v.1.3.x)

http://httpd.apache.org/dist/httpd/

MySQL

http://www.mysql.com/Downloads/MySQL-4.0/mysql-4.0.14.tar.gz

Bei MySQL entstehen ungefähr alle zwei Wochen neue Versionen. Deshalb hier noch die allgemeine Download Adresse (Achtung: Source-Files herunterladen, nicht Binaries):

http://www.mysql.com/downloads/

PHP

http://www.php.net/downloads.php

SSL

openSSL: http://www.openssl.org/source/

mod_SSL: http://www.modssl.org/source/

Erweiterungen

PDFlib: http://www.pdflib.com/pdflib/download/index.html

Zlib-Library: http://www.gzip.org/zlib/

Eigentliche Installation

Entpacken der Pakete

Die Pakete werden wie folgt entpackt:

```
tar xvfz paketename.tar.gz
```

Nun gibt es jeweils Verzeichnisse, in denen sich die entpackten Tarballs (.tar.gz Dateien) befinden.

MySQL

Wir installieren zuerst MySQL, da die Datenbank unabhängig von den anderen Paketen ist.

Das Verzeichnis, in welchem MySQL seine Datenbanken (`data`) ablegt, kann mit dem Configure- Parameter `localstatedir` angegeben werden. Wir haben uns hier für den Pfad `/usr/local/mysql/data` entschieden. Mit dem Parameter prefix geben wir das Installationsverzeichnis unserer MySQL Datenbank an. (z.B. `/usr/local/mysql/4.0.14/` für unsere aktuelle MySQL Version). Nach der Installation erstellen wir noch einen Softlink. Auf diese Weise findet man nach der Installation einer neueren Version, die aktuelle unter `current`.

Im MySQL Source Verzeichnis:

```
./configure --prefix=/usr/local/mysql/4.0.14/ --
localstatedir=/usr/local/mysql/data
make
make install
ln -s /usr/local/mysql/4.0.14 /usr/local/mysql/current
mkdir /usr/local/mysql
mkdir /usr/local/mysql/data
/usr/local/mysql/current/bin/mysqlinstall_db
/usr/local/mysql/current/bin/safe_mysqld -u mysqladmin &
```

SSL Installation

Damit wir mit SSL arbeiten können, müssen wir unser System zuerst mit der SSL Funktionalität ausrüsten. Deshalb installieren wir zuerst openSSL:

Im openSSL Source Verzeichnis:

```
./config--prefix=/usr/local/openssl/0.9.6j
make
make test
make install
ln -s /usr/local/openssl/0.9.6j /usr/local/openssl/current
```

Apache und mod_SSL

Wir erstellen nun eine lauffähige Version von Apache, welche mit Dynamic Shared Objects umgehen kann. Dies ist eine überaus praktische Technologie. DSOs werden auch Apache Module genannt und funktionieren in etwa analog zu den bekannten Linux Modulen. Man kann sie einbinden und muss nicht für jedes neue Modul den ganzen Apache neu übersetzen.

Wir patchen die Quellen des Apache Webservers mit dem mod_SSL Paket. Gleichzeitig wird die Apache Configure Funktion aufgerufen. Das `datadir` Verzeichnis beherbergt das Webserver Hypertext Document Root. Wir haben uns hier für `/usr/local/apache/htdocs` entschieden.

Im mod_SSL Source Verzeichnis (*nicht* im Apache Source Verzeichnis!):


```
./configure --with-apache=../apache_source_verzeichnis \
--with-ssl=../openssl_source_verzeichnis \
--prefix=/usr/local/apache/1.3.28 \
--datadir=/usr/local/apache/htdocs \
--enable-module=most \
--enable-shared=max \
--enable-module=ssl

make
```

Im Apache Source Verzeichnis:

Server-Zertifikat erstellen

Wir erzeugen nun ein Zertifikat zur späteren Verwendung von SSL-Sitzungen. Dieses wird von einer Certificate Authority (CA) unterschrieben. Wir legen die CA hier auch gleich mit an.

CA erstellen:

Im Apache Verzeichnis:

`make certificate TYPE=custom`

Signature Algorithm:	R
Country Name:	CH
State or Providence:	ZH
Locality Name:	Winterthur
Organisation Name:	PhPepperShop
Organisational Unit Name:	CA
Common Name:	PhPepperShop CA
Email Address:	ca@mydomain.com
Certificate Validity:	365
Certificate Version:	3

Nachdem nun der CA-Schlüssel erzeugt wurde, geht es ans Server-Zertifikat. Dabei ist zu beachten, dass der *Common Name* den Rechnernamen darstellt, unter welchem der Server später per SSL erreicht werden kann! (z.B. `https://myshop.mydomain.com`)

Country Name:	CH
State or Providence:	ZH
Locality Name:	Winterthur
Organisation Name:	PhPepperShop
Organisational Unit Name:	Webmaster
Common Name:	myshop.mydomain.com
Email Address:	info@mydomain.com
Certificate Validity:	365
Certificate Version:	3

Die Schlüssel sollen natürlich *verschlüsselt* werden, dies also bestätigen (zuerst CA, dann Server). Angemerkt sei hier noch, dass bei jedem Neustart des Apache das Passwort für den Server-Key eingegeben werden muss.

Apache Installation vervollständigen

```
make install
ln -s /usr/local/apache/1.3.28 /usr/local/apache/current
```

Im Apache Source Verzeichnis:

Damit der Apache auch mit SSL-Unterstützung startet, müssen wir noch einen kleinen Eintrag in der Datei: `/usr/local/apache/current/bin/apachectl` ändern:

```
start)
if [ $RUNNING -eq 1 ]; then
    echo "$0 $ARG: httpd (pid $PID) already running"
    continue
fi
#Original:
#if $HTTPD ; then
#gaendert, -DSSL eingefuegt:
if $HTTPD -DSSL; then
    echo "$0 $ARG: httpd started"
else
    echo "$0 $ARG: httpd could not be started"
    ERROR=3
fi
;;
```

Nun müssen noch alle SSL Einträge aus der httpd.conf.default in die Datei httpd.conf kopiert werden. Es geht v.a. darum, die Module `libssl` und `modssl` zu laden (LoadModule).

PHP

PHP ist ein komplexes System und damit es wie von uns erwartet funktioniert, müssen wir zuerst noch ein paar Hilfslibraries kompilieren und installieren:

PDFlib

Im PDFlib Source Verzeichnis:

```
./configure --prefix=/usr/local
make
make install
```

Zlib

Im Zlib Source Verzeichnis:

```
./configure
make
make install
```

PHP konfigurieren und kompilieren

Wichtig ist das `with-config-file-path`-Attribut. Im angegebenen Pfad wird PHP später nämlich seine `php.ini` suchen.

Im PHP Source Verzeichnis:

```
rm config.cache
./configure \
--with-apxs=/usr/local/apache/current/bin/apxs \
--with-mysql=/usr/local/mysql/current \
--with-zlib \
--enable-ftp \
--with-pdflib \
--enable-versioning \
--enable-track-vars=yes \
--enable-url-includes \
--enable-sysvshm=yes \
--enable-sysvsem=yes \
--with-config-file-path=/etc
make
make install
```

Nachdem nun PHP installiert wurde, müssen wir nun noch eine Datei ins etc-Verzeichnis kopieren:

```
cp ./php.ini-dist /etc/php.ini
```

Es empfiehlt sich, nun in der `php.ini` den *Safe Mode* einzuschalten. Dies macht man, indem man einen Editor (z.B. `pico`) nimmt und die Zeile sucht, in welcher safe_mode `= Off` steht und diesen dann auf `safe_mode = On` stellt. Die Register Globals sind standardmässig = Off. (Erst der PhPepperShop ab Version 1.4 kann mit dieser Einstellung korrekt umgehen.)

Konfiguration

Wir müssen im Verzeichnis `/usr/local/apache/current/conf/` die zentrale Konfigurationsdatei des Apache (`httpd.conf`) editieren. Zuerst sollten alle aktuellen Versionsnummern die im Skript vorkommen (bei uns z.B. 1.3.28) durch current ersetzt werden. Nun müssen wir noch die Mime-Types für PHP hinzufügen, beziehungsweise auskommentieren. Dies macht man am besten indem man folgende Zeile sucht, auskommentiert und .phtml noch hinzufügt:

```
AddType application/x-httpd-php     .php     .phtml
```

Zusätzlich konfigurieren wir Apache nun so, dass er auch PHP-Dateien als Default-Index Dateien akzeptiert. Dazu suchen wir den Eintrag DirectoryIndex und fügen noch die PHP Datei-Endungen hinzu:

```
DirectoryIndex    index.html    index.php    index.htm
```

Starten

Wir können nun den Apache Webserver starten, indem wir folgende Anweisung eingeben:

```
/usr/local/apache/current/bin/apachectl start
```

Wenn SSL korrekt installiert wurde, wird man an dieser Stelle angehalten seine Passphrase einzugeben. Danach startet der Apache Webserver.

