LinuX Enterprise

LinuX Enterprise

Deutschland € 5,80
Österreich € 6,90
Schweiz sFr 10,90

S&S Magazin für Professional Linux & Open Source Solutions

3.05

Auf CD

- Pyramid BenHur 2
- Intel C++ Compiler for Linux Version 8.1
- Intel Fortran for Linux Version 8.1
- Terrashop-Katalog
  Terrashop-Katalog im PDF-Format
- Kernel & Software
  Linux-Kernel 2.6.11.5
  ActiveState Perl 5.8.6.811
  Gimp 2.2.4
  Python 2.4
  Ruby 1.8.2

CPAN Perl-Module
Python-Tools
Ruby-Tools

CD-Inhalt auf Seite 3

Web Security

- Wie Sie Apache besser schützen
- Sicherheit aus dem Kernel mit SELinux
- Honeypots als Lockvögel

Boost Your Productivity

Die Zukunft von C++



Special Softwarepatente

Gefahr oder Chance?

ASP.NET mit Apache

Entwicklung von Webanwendungen unter Mono und Linux

www.linuxenterprise.de

SQLObject

Object Relational Mapping in Python

Highway to Heaven?

SCM- und CRM-Lösungen unter Linux

Editoren

VIM, Emacs und mcedit Editoren im Vergleich



Wie Sie Apache besser schützen

Auf Kriegspfad

Der beliebte HTTP-Server Apache ist ein waschechtes Open-Source-Produkt und die solide Basis für zahllose Webauftritte: Das Spektrum der Benutzer erstreckt sich vom Privatmann bis hin zu international bekannten Onlineshops. Allerdings ist nicht jeder Anwender mit allen Einzelheiten und Finessen der Administration vertraut. Diese Aussage gilt insbesondere für Fragen der Sicherheit, denn bereits eine saubere Installation erfordert Hintergrundwissen. Der Beitrag wird vor allem ausgewählte Punkte der Verwaltung aus dem Blickwinkel der Sicherheit berücksichtigen.

von Thomas Kaufmann

Die Fakten: Apache [1] war und ist eines der Flagschiffe der Open-Source-Bewegung, steht für alle gängigen Plattformen zur Verfügung (Unix, Linux, Mac und Windows) und wird derzeit noch in zwei „Produktlinien" vertrieben: die 1.3.x-Reihe sowie die 2.0.x-Reihe. Dass sich der freie Webserver bis zum heutigen Tag großer Beliebtheit erfreut, kommt nicht von ungefähr – es folgt eine Auflistung guter Gründe, warum Apache immer noch erste Wahl ist:

- Apache ist kostenlos.
- Apache ist ausgereifte Software.
- Apache ist leicht zu konfigurieren.
- Apache ist robust und zuverlässig.
- Apache ist durch Third-Party-Module erweiterbar.
- Apache wird aktiv weiterentwickelt.
- Apache ist nicht proprietär.
- Der Anwender erhält eine ausgezeichnete Dokumentation.

Im Verlauf der letzten zehn Jahre ist der Webserver den Kinderschuhen entwachsen und hat ordentlich an Umfang gewonnen. Die wenigsten Anwender haben allerdings die Zeit, die dicken Apache-Bibeln zu wälzen und sich mit allen Feinheiten der Administration zu beschäftigen. Der vorliegende Beitrag wird sich daher auf Teilaspekte der Installation und Konfiguration, der Verwendung von Modulen und last not least des Loggings konzentrieren.

Ein Wort in eigener Sache: Der Autor hat Apache 2.0.40 und Red Hat 9.0 verwendet. Die Pfade können deshalb abweichen. Schauen Sie in diesem Fall bitte in Ihre Dokumentation. Tipp: Wenn Sie nicht wissen, ob Apache auf Ihrer Maschine läuft, ist es möglich, die Version des Webservers in Erfahrung zu bringen (gilt wie gesagt für Red Hat):

```
[root@localhost apache]# /usr/sbin/apachectl -V
```

Die Ausgabe liefert außerdem andere nützliche Informationen, die aber aus Platzgründen weggelassen wurden:

```
Server version: Apache/2.0.40
Server built:  Feb 25 2003 05:01:56
...
...
```

Kontrolle ist besser

Grundsätzlich existieren zwei Möglichkeiten, um Apache zu verwenden: Entweder nutzen Sie das mitgelieferte Paket der Linux-Distribution (Apache ist unter Linux vorinstalliert). Das ist der einfache Weg, dafür fehlt Ihnen aber die volle Kontrolle über bestimmte Einstellungen, die Sie während der Kompilierungsphase hätten. Die andere Variante: Sie besorgen sich das Archiv aus dem Netz und installieren es selbst.

Wenn Sie Apache aus dem Netz abholen, ist übrigens nicht automatisch gewährleistet, dass Sie tatsächlich ein „sauberes" Paket erhalten. Diese Feststellung ist wichtig! Denn es ist durchaus denkbar, dass Hacker Pakete präpariert haben (beispielsweise auf Mirror-Sites). Klingt abenteuerlich? An dieser Stelle sei an Debian erinnert: Der Server des Debian-Projektes wurde vor einiger Zeit von Hackern kompromittiert und niemand konnte nach diesem Vorfall garantieren, ob nicht auch Teile der Debian-Sourcen betroffen waren (Stichwort „backdoor"). Soviel dazu. Diese realistische Gefahr ist von den Apache-Entwicklern eindeutig erkannt worden. Deshalb stehen auf der Website des Apache-Projekts nur Pakete zum Download bereit, die persönlich von einem der Entwickler mit einer digitalen Signatur, unter Verwendung von PGP und MD5, versehen worden sind.

```
httpd-2.0.53.tar.gz is signed by Justin Erenkrantz E2226795

httpd-1.3.33.tar.gz is signed by Jim Jagielski 08C975E5
```

Unter [2] finden Sie detaillierte Erläuterungen und weiterführende Links. Sie haben nun die Möglichkeit, entweder mit *GnuPG* [3], *md5* oder *md5sum* (rufen Sie *man md5sum* auf) die Verifizierung durchzuführen. Nach Überprüfung der Integrität der Daten folgt der nächste Schritt, die Linux-typische Installation (bitte als root durchführen). Zuerst wird der Tarball entpackt:

```
tar xvzf httpd-2.0.xxx.tar.gz
```

Danach wechseln Sie in das neu erstellte Verzeichnis mit *cd httpd-2.0.xxx*. Mit Hilfe des Schalters *--prefix* wird das Installationsverzeichnis bestimmt:

```
./configure --prefix=/usr/local/apache --enable-mods-
                                                shared=most
```

Die Option *--enable-mods-shared=most* sorgt dafür, dass nur Standardmodule kompiliert werden. Module mit experimentellem Status (*mod_mime_magic*, *mod_cern_meta*, *mod_user_track*, *mod_unique_id*) werden hingegen nicht berücksichtigt. Daraufhin folgt der Aufruf von *make* und *make install*. Wenn sich Apache in */usr/local/ apache* befindet, können Sie die Installation abschließend mit

```
tail -f /usr/local/apache/logs/error_log
```

testen. Nähere Angaben für spezielle Optionen entnehmen Sie bitte dem beiliegendem Readme.

Die Konfiguration checken

Es gibt einige Dateien, die Sie im Zusammenhang mit Apache unbedingt kennen müssen, wenn der Server reibungslos und zuverlässig seinen Dienst verrichten soll (in *man httpd* finden Sie am Ende die Pfade):

```
/etc/httpd/conf/httpd.conf
/etc/mime.types
/etc/httpd/conf/magic
/var/log/httpd/error_log
/var/log/httpd/access_log
/var/run/httpd.pid
```

Die Datei *httpd.conf* ist gewissermaßen die Schaltzentrale für die Administration des Webservers. Diese besitzt einen logischen Aufbau und ist in Sektionen und Abschnitte unterteilt. Dort befinden sich sämtliche Angaben (nebst ausführlichen Kommentaren) zur Steuerung und Kontrolle des Servers. Falls Sie nicht sicher sind, in welchem Verzeichnis sich *httpd.conf* befindet, können Sie diesen Befehl absetzen: *locate httpd.conf*

Besonders wichtig sind in diesem Kontext auch *error_log* und *access_log*. Beide Dateien registrieren entweder Fehler oder Anfragen von Clients. Für das Logging und besonders für das Debuggen von Webanwendungen sind sie unverzichtbar.

Die Datei *httpd.pid* enthält wiederum die Prozess-ID des Servers (die interne Bezeichnung für Apache lautet *httpd*; das abschließende kleine „d“ steht für Daemon). Es ist manchmal nützlich, die PID (Process ID) zu kennen – wenn Sie beispielsweise den Serverprozess einmal gewaltsam beenden müssen. Eine PID, das gilt für jeden Prozess unter Linux, bringen Sie mit diesem Kommando in Erfahrung:

```
[ich@localhost apache]$ ps ax | grep httpd
2303 ?    S    0:00 /usr/sbin/httpd
2320 ?    SW   0:00 [httpd]
...
...
```

Die Apache-PID lautet hier also *2303*.

Nun ein Beispiel für die Konfiguration mit *httpd.conf*. Öffnen Sie die Datei und suchen Sie die Zeichenkette „ServerTokens“ (für Änderungen in *httpd.conf* sind root-Rechte erforderlich). Über ServerTokens justieren Sie die Ausgabe der Kennung des Webservers.

```
# Don't give away too much information about all the
                                                subcomponents
# we are running.  Comment out this line if you don't mind
                                                remote sites
# finding out what major optional modules you are running
ServerTokens Full
```

Der Kommentar sagt bereits alles: Meistens ist es eine sehr gute Idee, möglichst wenig Information über Ihre Infrastruktur nach „draußen“ dringen zu lassen. Mit der Einstellung „ServerTokens Full“ liefert Apache definitiv zuviel Information an einen Client. Denken Sie nicht, das wäre eine Bagatelle, da Angreifer, bewaffnet mit Server-Scannern wie „Whisker“ [4] und „Nikto“ [5], möglichst viel Wissenswertes über Ihren Webserver (inklusive Website) in Erfahrung bringen möchten.

Um dem Beispiel folgen zu können, besorgen Sie sich entweder Whisker oder Nikto. Die Installation ist leicht, Sie brauchen lediglich eine aktuelle Perl-Installation, und die ist unter Linux immer an Bord. Der Autor hat für den Test Nikto verwendet – hier der Aufruf (der Schalter *-h* gibt den Host an):

```
perl nikto.pl -h 192.168.0.102
```

Die Ausgabe (aus Platzgründen nur auszugsweise):

```
[ich@localhost]$ perl nikto.pl -h 192.168.0.102
-***** SSL support not available (see docs for SSL install
                                                instructions) *****
---------------------------------------------------------------------
- Nikto 1.34/1.29   -   www.cirt.net
+ Target IP:    192.168.0.102
+ Target Hostname: 192.168.0.102
+ Target Port:   80
+ Start Time:    Mon Mar 7 23:18:02 2005
---------------------------------------------------------------------
- Scan is dependent on "Server" string which can be faked,
                                                use -g to override
+ Server: Apache/2.0.40 (Red Hat Linux) mod_ssl/2.0.40
                                                OpenSSL/0.9.7a DAV/2
...
```

Interessant ist an dieser Ausgabe Folgendes: Wie Sie sehen, liefert Apache auf Anfrage die Zeichenkette *Apache/2.0.40 (Red Hat Linux) mod_ssl/2.0.40 OpenSSL/0.9.7a DAV/2*. Das ist schlecht! Mit Hilfe eines Portscans, durchgeführt mit „nmap“ [6], lässt sich dann wahrscheinlich sogar noch die Version des Betriebssystems in Erfahrung bringen, was eigentlich noch schlechter ist. Somit hätte ein potenzieller Angreifer bereits Wesentliches erfahren. Der nächste Schritt wäre, sich aus einschlägigen Quellen (wie zum Beispiel Bugtraq, Hacker-Sites, etc.) weitere Informationen über Schwachstellen und passende Exploits (Programme, die gezielt Schwachstellen ausnutzen) zu besorgen. Den Rest kennt man.

Zum Glück lässt sich die Geschwätzigkeit des Apachen unterbinden. Ändern Sie die Standardeinstellung von ServerTokens in „ServerTokens Prod“. Achtung: Sie müssen den Server wahrscheinlich neu starten, damit die Änderungen auch wirksam werden:

```
[root@localhost conf]# /usr/sbin/apachectl restart
```

Lassen Sie Nikto nun abermals von der Leine (siehe Aufruf oben). Das neue Resultat sieht so aus:

```
- Scan is dependent on "Server" string which can be faked,
                                    use -g to override
+ Server: Apache
```

Schon besser. Lesen Sie in *httpd.conf* den Kommentar zu ServerTokens und experimentieren Sie mit den Einstellungen – die Option *Full* liefert die meisten Informationen, *Prod* die wenigsten (außerdem gibt es noch weitere Attribute wie *OS*).

Des Weiteren ist es eine gute Idee, die Standard-CGI-Skripts *printenv* und *testcgi* aus dem *cgi-bin*-Verzeichnis zu entfernen (diese Aussage gilt vor allem für Produktionsserver). Beide Dateien waren in der Vergangenheit für Sicherheitsprobleme verantwortlich und sollten besser gelöscht beziehungsweise in einen neutralen Ordner verschoben werden. Wenn wir schon dabei sind: Im gleichen Zuge sollten Sie die Default-HTML-Seiten aus */htdocs* entfernen, die dort meist in mehreren Sprachen vorhanden sind. Ersetzen Sie sie vielmehr durch Ihren eigenen Webauftritt.

Erweiterungen

Für den Apache-Server existieren etliche Third-Party-Module: *mod_php*, *mod_perl* und *mod_python* sind Ihnen vielleicht schon einmal aufgefallen. Vereinfacht gesagt bohren Third-Party-Module den Aktionsradius des Webservers beträchtlich auf und stellen dem Anwender eine spezielle Funktionalität zur Verfügung. Beispielsweise erhalten Perl-Entwickler durch die Verwendung von *mod_perl* verbesserte Performance und Support für CGI-Anwendungen (die Aussage gilt selbstverständlich ebenso für *mod_php*). Ein paar Module, die Sie kennen sollten:

- *mod_hackdetect*
- *mod_hackprotect*
- *mod_security*

Unter [7] finden Sie eine Übersicht über die derzeit verfügbaren Third-Party-Module für Apache. Leider ist nicht jedes Third-Party-Modul frei verfügbar. Die beiden erstgenannten Module sind kommerzielle Produkte, dennoch sind sie empfehlenswert. Das Modul *mod_security* finden Sie unter [8]; es wird von Ivan Ristic betreut und obendrein kostenlos abgegeben.

Wozu ist *mod_security* gut? Im Prinzip handelt es sich um einen effizienten Request-Filter, d.h. mit anderen Worten, Anfragen von Clients werden auf verdächtig aussehende URLs untersucht. Eine umfassende Beschreibung von *mod_security* sprengt den Rahmen des Artikels, weshalb nur ein paar grundlegende Einstellungen gezeigt werden. (Achtung: Sie müssen dieses Modul extra herunterladen, denn es gehört nicht zum offiziellen Apache-Paket.)

Bei *mod_security* wird grundsätzlich auf zwei Ebenen konfiguriert: Einmal werden globale Settings vorgenommen, dann erfolgt die Feineinstellung für den Filter. Nach der Installation von *mod_security* enthält die Datei *httpd.conf* einen neuen Eintrag:

```
LoadModule security_module   modules/mod_security.so
```

Überdies müssen Sie selbst Hand anlegen, *httpd.conf* modifizieren und anschließend Apache neu starten. Legen Sie zuerst eine neue Sektion an, die nur ausgelesen wird, wenn das Modul tatsächlich aktiv ist. Dadurch erspart man sich später das Auskommentieren langer Passagen. Hier die umgebenden Tags:

```
<IfModule mod_security.c>
...
</IfModule>
```

Die in der Folge genannten Optionen gehören sämtlich zwischen diese Tags. Zuerst wird die Filter-Engine *SecFilterEngine On* aktiviert. Und die nächste Einstellung *SecFilterCheckURLEncoding On*. Ein Wort zu dieser Einstellung: Bei der URL-Kodierung werden manche Zeichen, Steuerzeichen und bestimmte alphanumerische Symbole durch das %-Symbol gefolgt von zwei hexadezimalen Zahlen kodiert (es handelt sich genau genommen um die Hex-Kodierung des ASCII-Wertes eines Zeichens).

Für die hexadezimale Kodierung sind die Werte *0* bis *F* erlaubt; eine Zeichenkette wie „Hello World“ wird somit in *Hello%20World%21* umgewandelt. Der Hex-Wert *21* entspricht dem ASCII-Wert *33* (in der ASCII-Tabelle finden Sie unter *33* das Ausrufezeichen). Auch diese Angabe wäre korrekt: *%2F%A3*

Das Modul *mod_security* hält nun Ausschau nach unvollständiger beziehungsweise „fehlerhafter“ URL-Kodierung. Ein Beispiel für unvollständige URL-Kodierung ist *?param=val%A*

Am Ende fehlt ein Zeichen (es müssen zwei Hex-Zahlen sein). Schauen Sie sich diese Zeile an: *?param=val%AX*. Auch in diesem Fall schlägt *mod_security* Alarm, da die Vermutung nahe liegt, dass jemand Requests manipuliert. Solche Techniken werden mitunter für Umgehungsstrategien angewendet. Wie Sie sehen konnten, bietet das Modul nützliche Funktionalität, die den Server absichern hilft. Weitere In-

Anzeige

formationen finden Sie in der Dokumentation von *mod_security*.

Eine Warnung zum Schluss: Third-Party-Module sind im Grunde eine gute Sache und helfen, die Funktionen des Webservers sinnvoll zu erweitern. Allerdings verursachen manche Module durchaus ernsthaften Schaden. Dazu muss man wissen, dass viele Module nicht von den Apache-Entwicklern stammen. Vielmehr werden dieselben von freien Entwicklern beigesteuert, die eine bestimmte Erweiterung benötigen. Das kann leider bedeuten, dass die Qualität solcher Produkte, vorsichtig formuliert, zweifelhaft ist. Sie sollten vor einer Installation sorgfältig prüfen, ob Sie erstens Modul *mod_xyz* tatsächlich benötigen und zweitens genau hinschauen, von welcher Site Sie ein Modul herunterladen.

Das Gedächtnis des Indianers

Logging ist eingedenk der aktuellen Bedrohungen besonders wichtig und muss heutzutage nicht weiter begründet werden. Apache unterscheidet zwei Logging-Typen: Fehler-Dateien (*error_log*) und Transfer-Logs (*access_log*). Die letztgenannte Datei ist übrigens, im Gegensatz zu *error_log*, nicht zwingend notwendig und muss vom Anwender aktiviert werden (meistens nur bei einer Neu-Installation).

Zuerst ein paar Worte zu *error_log*: Diese Datei befindet sich immer im gleichen Verzeichnis *<ServerRoot>/logs/error_log*. Sie können das Format von *error_log* nicht beeinflussen, wohl aber den Speicherort neu bestimmen. Dazu dient die *ErrorLog*-Direktive in *httpd.conf*:

```
ErrorLog /var/log/httpd/error_log
```

Und falls Sie besonders abenteuerlustig sind, dürfen Sie auch Folgendes tun (wovon im Normalfall nur dringend abgeraten werden kann ;-)): *ErrorLog /dev/null*. Es existieren zudem verschiedene Loglevel: *emerg*, *alert*, *crit*, *error*, *warn*, *notice*, *info* und *debug*. Standardmäßig ist in *httpd.conf* für *LogLevel* die Option *warn* aktiviert. Mit *debug* erhalten Sie hingegen die meisten Meldungen. Das führt allerdings schnell zu riesigen Logfiles und beeinträchtigt wahrscheinlich die Übersichtlichkeit in *error_log*.

Die Einstellung *LogLevel debug* ist dann nützlich, wenn Sie neue Features (neue Module etc.) testen; für Produktionssysteme wird sie nicht empfohlen. Experimentieren Sie mit den Optionen und finden Sie die passende Einstellung heraus.

Und noch eine interessante Möglichkeit: Es ist möglich, die Fehlerausgabe an *syslogd* umzuleiten. Das erreichen Sie mit dieser Einstellung: *ErrorLog syslog*. Schauen Sie für weitere Informationen auch in die Manpage *man syslogd*.

Kommen wir nun zum zweiten Typ, den Transfer-Logs. In *httpd.conf* wird die Datei *access_log* auf diesem Wege aktiviert:

```
TransferLog /var/log/httpd/access_log
```

Dies muss normalerweise der Anwender eingeben. Bei einem vorinstalliertem Apache wird dieser Eintrag aber sehr wahrscheinlich bereits in *httpd.conf* stehen. Falls hinter *TransferLog* ein relativer Pfadname steht, wird derselbe an *<ServerRoot>* angefügt. Angenommen, die Apache-Installation residiert dann unter */usr/local/apache*:

```
TransferLog /usr/local/apache/logs/access_log
TransferLog logs/access_log
```

Beide Zeilen sind somit identisch. Lassen Sie uns einmal nachschauen, wie ein typischer Eintrag in *access_log* aussieht. Dazu suchen wir nach Anhaltspunkten für den (weiter oben durchgeführten) Nikto-Scan. Sie können *access_log* mit Hilfe von *grep* gezielt nach Zeichenketten durchsuchen. Der Aufruf:

```
[root@localhost apache]# grep -i nikto /etc/httpd/logs/
                                        access_log | less
```

Und das Resultat (die Ausgabe wurde aus Platzgründen gekürzt):

```
192.168.0.102 - [07/Mar/2005:22:39:20] "HEAD /
        HTTP/1.1" 200 - "-" "Mozilla/4.75 (Nikto/1.34)"
192.168.0.102 - [07/Mar/2005:22:39:20] "HEAD /
        HTTP/1.1" 200 - "-" "Mozilla/4.75 (Nikto/1.34)"
```

Apache bietet mit *error_log* und *access_log* zwei verschiedene Formen des Recordings. Von den *error_log*-Aufzeichnungen profitieren besonders Administratoren; aber auch Webentwickler, wenn sie CGI-Anwendungen debuggen. In der Regel stehen dort Meldungen zu Performance-Engpässen und anderen Fehlern. Wenn nun also etwas nicht wie gewohnt funktioniert, schauen Sie zuerst in *error_log* nach. Mit *access_log* sind Sie stets informiert, welcher Client sich mit Ihrem Server verbinden möchte und auch ausgehende Verbindungen werden aufgezeichnet.

Fazit

Apache hat zweifellos seinen Stammplatz im Internet behauptet: Das hat dann zur Folge, dass man lernen muss, den Server effektiv zu schützen, denn die Standardeinstellungen reichen meistens nicht mehr aus. Und Sie müssen stets auf dem Laufenden bleiben, sonst gibt es ganz schnell ein böses Erwachen. Erfreulicherweise bietet Apache mannigfaltige Möglichkeiten für die sichere Konfiguration. Und mithilfe von Third-Party-Modulen erweitern Sie den Funktionsumfang sogar noch beträchtlich.

Zudem können eine restriktive Sicherheitspolitik und systematisches Logging die Sicherheit einer Website weiterhin erheblich verbessern. Fakt ist, dass die Administration eines Apache-Servers kein Kinderspiel ist. Trotzdem: Mit wachem Auge für die aktuellen Entwicklungen im Netz haben Sie gute Karten im Wettlauf mit den bösen Jungs (und Mädchen). ■

Links & Literatur

[1] Apache-Projekt: www.apache.org

[2] Apache-Sourcen: httpd.apache.org/download.cgi

[3] GnuPG: www.gnupg.org

[4] Whisker: www.wiretrip.net/rfp

[5] Nikto: www.cirt.net/code/nikto.shtml

[6] nmap: www.insecure.org

[7] Third-Party-Module für Apache: modules.apache.org

[8] mod_security: www.modsecurity.org

[9] Infos zu IT-Sicherheit: www.sans.org

[10] Peter Wainright: Pro Apache. apress, 2004

[11] Tony Mobily: Hardening Apache. apress, 2004