



Den Apache 2 Webserver absichern v1.0

Für das
Digital-Library Projekt
- digital-library.de.vu -

Erstellt von der
Parallel Minds Cooperation
- www.paramind.info -

Geschrieben von [CONVEX]

Vorwort:

Es wird immer noch der Fehler gemacht Sicherheit als einen Endzustand zu betrachten. Sicherheit ist jedoch kein Endzustand, sondern ein fortwährender Prozess. Sicherheitsstrukturen und Einstellungen sind ständig zu prüfen und den aktuellen Gegebenheiten anzupassen. Gegebenheiten die anfangs als sicher eingestuft wurden, können sich innerhalb kürzester Zeit ändern. Vor allem im IT-Bereich ist dies zu beachten.

Dieses Dokument befasst sich hauptsächlich mit den Möglichkeiten den Apache 2 Webserver abzusichern. Es wird eine kompakte Auflistung von Einstellungsmöglichkeiten geliefert, auf die ein verantwortungsbewusster Administrator einzugehen hat.

Die hier vorgestellten Konfigurationen sind in erster Linie für Apache Webserver gedacht, die in einem Firmennetzwerk eingesetzt werden. Aber auch für den privaten Einsatz lassen sich verwertbare Informationen ableiten.

Den vorgenommenen Einstellungen liegen Empfehlungen aus verschiedenen Quellen zugrunde:

- Sicherheitsstudie zum Apache Webserver vom Bundesamt für Sicherheit in der Informationstechnik (BSI).
- Apache-Konfigurationsanweisungen (<http://httpd.apache.org/docs/>)
- Apache Security - The Complete Guide To Securing Your Apache Web Server (Herausgegeben von O'Reilly).

Grundsätzliche Vorgaben

Das BSI weist darauf hin, dass neben Angriffen auf die Webserver-Applikation auch Angriffe auf Schwachstellen des verwendeten Betriebssystems oder anderer über das Netz erreichbarer Applikationen möglich sind. Hierzu müssen u. a. folgende Punkte beachtet werden:

- Es sollten immer aktuelle Software-Releases des Betriebssystems und aller anderen auf dem Webserver installierten, sicherheitsrelevanten Applikationen verwendet werden. Alle sicherheitsrelevanten Patches oder Bugfixes sind einzuspielen.
- Die Installation des Betriebssystems sollte möglichst "minimal" sein, so dass nur die notwendigen Dienste und Programme laufen.
- Die Rechte, über die der Webserver oder andere Server-Software verfügen, sollten möglichst restriktiv vergeben werden. Dadurch kann der Schaden begrenzt werden, der entsteht, falls eine Sicherheitslücke in einem der Programme gefunden und von einem Angreifer ausgenutzt werden sollte.
- Zusatzsoftware, die durch den Webserver gestartet wird (wie CGI-Skripten oder Ähnliches), sollte sorgfältig auf sicherheitsrelevante Fehler überprüft werden.

Um auf potentielle Sicherheitsvorfälle angemessen reagieren zu können wird das Prinzip von Kontrolle und Beobachtung eingesetzt u.a. mit Hilfe von

Protokolldateien. Nur so können Sicherheitsvorfälle geprüft und eine Revisionsicherheit gewährleistet werden.

Allgemeine Sicherungsmaßnahmen für Apache 2

Start des Apache-Webservers

Der Apache 2 Serverprozess sollte aus den Startup-Skripts des Betriebssystems heraus erfolgen, damit auch nach einem Reboot der Webserver direkt zur Verfügung steht. Werden zudem noch SSL-Erweiterungen genutzt, so kann es unter Umständen dazu kommen, dass ein Administrator zur Freigabe der privaten Schlüssel für das SSL-Zertifikat manuell eine Passphrase eingeben muss. Hier wäre dann ein automatischer Neustart nicht möglich.

Damit der Apache-Webserver den für den Dienst WWW vorgesehenen Standard-Port 80 benutzen kann, weist das BSI darauf hin, dass der Apache-Webserver mit *root*-Berechtigung gestartet werden muss. Für den Apache-Webserver sollte jedoch unbedingt ein eigenes Benutzerkonto und eine eigene Gruppe, beispielsweise *wwwrun*, angelegt werden. Der Webserver sollte (mittels der Direktiven *User* und *Group*) nach dem Start in diesen Sicherheitskontext wechseln.

Zugriffsrechte auf Dateien und Verzeichnisse

Es sollte bei der Konfiguration der Dateizugriffsrechte dahingehend geachtet werden, dass nur dem Benutzer *root* und einer entsprechenden Systemgruppe das Apache-Verzeichnis und alle darüber liegenden Verzeichnisse gehören.

Nur der Administrator darf auf diese Verzeichnisse Schreibzugriff haben. Dies gilt auch für alle Unterverzeichnisse des Apache-Verzeichnisses, in denen sich Binärdateien, Konfigurations- oder Logdateien befinden. In der Installation einer Quellcode-Distribution sind dies die Verzeichnisse *bin*, *conf* und *logs*. Auch die Binärdateien selbst sollten nur von *root* geschrieben werden können.

Freigegebene Ports

Die Datei „*/etc/apache2/listen.conf*“ enthält Angaben zur IP des Servers und den verwendeten Ports. Sie sollte den individuellen Bedürfnissen entsprechend angepaßt werden. In Regel ist hier der Standardport 80 eingetragen und bei SSL Unterstützung für verschlüsselte Zugriffe der Port 443. Nicht gewünschte Freigaben werden in dieser Liste auskommentiert bzw. gelöscht.

Geladene Module

Dem Sicherheitsgrundsatz folgend, sollte man darauf achten, dass die Konfigurationen und die vom Webserver gelieferten Informationen auf das notwendigste beschränkt werden. Das automatische Senden bei servergenerierten Seiten der Apache Versionsinformation (z.B. Fehlermeldungen) wird über:

```
APACHE_SERVERSIGNATURE="off"
```

in der Konfigurationsdatei `/etc/sysconfig/apache2` ausgeschaltet. In dieser Datei können auch nicht benötigte Module aus der Apache Konfiguration entfernt werden. Ein Beispiel einer solchen Modul-Konfiguration könnte so aussehen:

```
APACHE_MODULES="access actions alias asis auth autoindex cgi dir imap include  
log_config mime negotiation setenvif status userdir ssl php5"
```

Aus der Default-Liste der zu ladenden Module werden die rot markierten herausgenommen (außer bei explizitem Wunsch) und die grün markierten gegebenenfalls eingetragen.

Eine Beschreibung der Module findet sich im Bedarfsfalle unter:

<http://httpd.apache.org/docs/2.0/mod/>

Es sollten grundsätzlich nur die Module im Server aktiviert werden (das geschieht in der Konfigurationsdatei bei dynamischen Modulen bzw. beim Übersetzen des Quelltextes bei fest einkompilierten Modulen), die für den Betrieb des Webserver unbedingnt benötigt werden.

Es empfiehlt sich die Module *autoindex*, *userdir* und *status* aus der Default-Liste zu entfernen.

- Über *autoindex* werden Indexlisten von Ordnern ohne `index.html` ausgegeben, wenn dies nicht in der Serverkonfiguration ausgeschaltet wird. Solange dies nicht explizit gewünscht wird sollte das Modul entfernt werden. Ansonsten bestände die Gefahr, dass Dateien in einem Webverzeichnis, die nicht zur Veröffentlichung bestimmt sind, zugänglich gemacht werden.
- Auch Module wie *status* oder *info* sollten nach Möglichkeit deaktiviert werden, da bei Verwendung dieser Module der Apache-Webserver eine Reihe von Statusinformationen über den Webserver über die HTTP-Schnittstelle bereitstellt. Ansonsten eröffnet sich die Möglichkeit, von außen Details über die Konfiguration des Webserver in Erfahrung zu bringen, die dann als Grundlage für einen weiterführenden Einbruchsversuch dienen. Wird aber ein Zugriff auf die Statusinformationen gewünscht, so sollte zumindest eine sehr restriktive Zugangsbeschränkung über entsprechende `<location>`-Direktiven vorgenommen werden. Ist kein Datentransfer von Clients zum Webserver vorgesehen, so sollten hier speziell PUT-Requests nicht freigegeben werden.

Ein weiterer Hinweis vom BSI verweist auf die Option *FollowSymLinks* des Webserver, die ausgeschaltet werden sollte, da sonst über symbolische Links Dateien von außerhalb der *DocumentRoot* über den Web-Dokumentenbaum

zugreifbar werden. Gegebenfalls kann stattdessen die Option *SymLinksIfOwnerMatch* verwendet werden. Das Überschreiben der Konfigurationseinstellungen durch Einträge in den *.htaccess*-Dateien sollte so weit eingeschränkt werden, wie dies die Anforderungen des Apache-Einsatzes zulassen.

mod_security einsetzen (optional)

Der Einsatz des Moduls *mod_security* ist optional. Das Modul stellt die Funktionen einer Application-Level-Firewall zur Verfügung und ermöglicht in den zu verarbeitenden Verbindungen nicht nur eine Filterung nach IP-Adressen, sondern ermöglicht auch eine Filterung nach Variablen-Parametern von PHP-Skripten, Cookie-Namen und Informationen in HTTP-Headern. Sie schützt damit vor bekannten als auch vor unbekanntem Angriffen. Der Einsatz dieses Moduls ist durchaus empfehlenswert aber nicht zwingend. Erwägt man den Einsatz dieses Moduls, dann sind ausführliche Tests der aufgestellten Regeln vor dem Produktiveinsatz dringend notwendig. Zudem verursachen diese Prüfungen einen gewissen Performanceverlust. Die Notwendigkeit einer solchen Sicherheitserweiterung muss also im Vorfeld gut überlegt sein.

Verwendung von CGIs durch ScriptAlias Direktive

Aufgrund der Möglichkeit, dass ausführbare Dateien eingeschleust werden können, sollte deren Ausführung auch nur in bestimmten, kontrollierten Bereichen ermöglicht werden. Die Ausführung kann durch eine der hier genannten Methoden eingeschränkt werden:

- Durch Verwendung von ScriptAlias Direktiven
- Per Konfiguration
- Durch Server-Side Includes
- Durch Zuweisung von Handlern, Types oder Filtern

Die Verwendung von „ScriptAlias“ macht eine schnelle und einfache Umsetzung der Vorgaben möglich:

Beispiel:

```
ScriptAlias /cgi-script/ „/usr/local/apache/cgi-bin/
```

Diese Direktive erzeugt einen virtuellen Webordner und erlaubt dort die Ausführung von CGI-Skripten ohne die Konfiguration des aktuellen Ordners zu verändern. Alternativ kann die Ausführung von CGI-Skripten auch per Konfiguration explizit definiert werden.

Um ein Verzeichnis als CGI-Verzeichnis zu deklarieren, ist auch das Setzen des Handlers „cgi-script“ möglich. Der Handler wird über die Konfigurationsanweisung *SetHandler* aktiviert. Zusätzlich muss mit der Options-Anweisung die Ausführung von CGI-Skripten für dieses Verzeichnis erlaubt werden.

Beispiel:

```
<Directory /usr/local/apache/cgi-bin>  
AllowOverride None  
Options ExecCGI  
SetHandler cgi-script  
</Directory>
```

Basiskonfiguration

Der Zugriff auf alle Dateien des lokalen Rechners über die HTTP-Schnittstelle sollte in der Apache-Konfigurationsdatei über einen *Limit* Abschnitt für das Wurzelverzeichnis / verhindert werden.

Lediglich der Zugriff auf das Verzeichnis mit den Webdokumenten (z. B. */usr/local/apache/htdocs*) sollte wieder mittels eines entsprechenden *Limit* Abschnittes für eingehende HTTP-Requests wie GET oder HEAD geöffnet

Installation des Apache-Webservers in einem *chroot*-Behälter (optional)

Unter Unix-Systemen gibt es noch die Möglichkeit einen Server mit *chroot* (change root) abzusichern. *chroot* ist ein Systemaufruf unter Unix, der ein Programm in seinem Zugriff auf einen Teil des Dateibaums beschränkt. Dies geschieht dadurch, dass alle Zugriffe, die dieses Programm (und die von ihm aufgerufenen Programme) auf das Dateisystem durchführt, relativ zu dem Verzeichnis erfolgen, das beim Aufruf der Funktion *chroot* angegeben wurde. Dieses Verzeichnis wird so zur Wurzel eines virtuellen Dateibaums, welches man auch als *chroot*-Behälter oder *chroot jail* bezeichnet.

Es steht neben dem Systemaufruf *chroot* steht auch ein ausführbares Programm mit dem gleichen Namen zur Verfügung, das zum Start beliebiger Programme in einem solchen *chroot*-Behälter genutzt werden kann. Der *chroot*-Mechanismus bietet zusätzliche Sicherheit, die auch dann noch wirksam ist, wenn die im Apache-Webserver implementierten Sicherheitsmechanismen versagen: Durch eine (nach heutigem Kenntnisstand für die aktuelle Version des Apache-Webservers hypothetische) Sicherheitslücke könnte es einem Angreifer gelingen,

- den Apache-Webserver dazu zu bringen, Daten auszuliefern, die dieser entsprechend seiner Konfigurationsanweisungen nicht ausliefern dürfte.
- oder sogar eigenen Programmcode im Prozessraum des Webservers auszuführen.

In diesen Fällen begrenzt der *chroot*-Behälter den potentiellen Schaden, da der Angreifer in beiden Fällen nur Zugriff innerhalb des *chroot*-Behälters erhält.

Ein weiterer Vorteil des *chroot*-Mechanismus besteht im zusätzlichen Schutz gegen eine Fehlkonfiguration des Apache-Webservers. Wenn Benutzern des Apache-Webservers zu weitgehende Zugriffsrechte eingeräumt wurden, bietet der *chroot*-Behälter hier zusätzliche Sicherheit. Der Apache-Webserver kann nur solche Dateien ausliefern, die sich innerhalb des ihm zugänglichen Verzeichnisbaums befinden, der hier auf den *chroot*-Behälter begrenzt ist.

Die Einrichtung einer chroot-Umgebung ist mit einem nicht unerheblichen Aufwand verbunden. Alle für den Betrieb des Webservers benötigten Dateien, einschließlich der benutzten Bibliotheken und der von diesen Bibliotheken benutzten Konfigurationsdateien, müssen in den *chroot*-Behälter kopiert werden. Es muss ermittelt werden, welche Dateien für den Betrieb des Webservers notwendig sind. Zudem reicht es nicht aus, diese Dateien einmal in den *chroot*-Behälter zu kopieren: Bei jeder Änderung des Betriebssystems durch Update oder Konfiguration müssen ggf. auch davon betroffene Dateien im *chroot*-Behälter angepasst werden.

Sollten bereits durch andere Sicherheitseinstellungen die Zugriffe auf externe Verzeichnisse beschränkt worden sein, ist die Verwendung eines Chroot-Umgebung als optional anzusehen.

Virtuelle Hosts

Unter SuSE-Linux finden sich Templates zum Erzeugen von virtuellen Hosts im Verzeichnis */etc/apache2/vhosts.d/*. Dort werden auch sämtliche Server als virtuelle Hosts abgelegt. Innerhalb der VirtualHostdateien muss auf eine möglichst restriktive Einstellung des DocumentRoot eines Virtual Hosts geachtet werden.

Beispiel:

```
<Directory />  
Order, Deny,Allow  
Deny from all  
AllowOverride None  
</Directory>
```

Dies stellt sicher, dass der Webserver seinen DocumentRoot Pfad nicht verlassen kann. In den einzelnen Virtual-Hosts muss sichergestellt werden, dass diese Einstellungen nur für benötigte Verzeichnisse durch die „Tags“ Location und Directory überschrieben werden. Die letzte Zeile regelt den Einsatz von *.htaccess*-Dateien zentral in der jeweiligen VirtualHost Konfiguration, da diese hier ebenfalls nur für spezielle Verzeichnisse zur Nutzung freigegeben werden müssen.

Fehlermeldungen

Standardmässig werden Fehlermeldungen (wie 404 – Datei nicht gefunden, usw.) durch sprachabhängige HTML-Seiten unter */usr/share/apache2/error* dargestellt. Diese Fehlermeldungen geben jedoch auch Informationen über das verwendete System und die Apache-Version preis.

Um solche Angaben zu unterdrücken, werden in der Datei

```
/usr/share/apach2/error/include/bottom.html
```

die entsprechenden Zeilen auskommentiert oder herausgelöscht.

Apache über Syslog-Host loggen (optional)

Das Bundesamt für Sicherheit in der Informationstechnik schlägt vor, Manipulationen von Einträgen in Systemprotokolldateien, insofern entgegenzuwirken, das man die Syslog-Daten nicht lokal verarbeitet, sondern direkt an einen externen Syslog-Dienst auf einem anderen Rechner (einem so genannten *Syslog-Host*) sendet. Unter Windows NT steht eine entsprechende Möglichkeit nicht zur Verfügung. Zur Einrichtung eines Syslog Hosts unter Unix müssen in der Datei */etc/syslog.conf* die Nachrichten des Apache-Webservers (deren Quelle und Priorität in der Konfigurationsdatei des Apache-Webservers festgelegt werden) auf den Syslog-Host weitergeleitet werden.

Je nach Einsatz des Apache-Webservers ist auch eine DNS-Anbindung nötig. Beispiele hierfür wären die Protokollierung von HTTP-Zugriffen mit dem Hostnamen anstelle der numerischen IP-Adresse oder die Konfiguration von Zugriffsbeschränkungen auf den Webserver über symbolische Hostnamen. Die Protokollierung von Hostnamen anstelle der numerischen IP-Adresse kann z. B. in einem Intranet sinnvoll sein, in dem IP-Adressen dynamisch über DHCP vergeben werden, während die symbolischen Hostnamen einen Rechner dauerhaft identifizieren. *Die Einrichtung eines Syslog-Hosts ist mit erheblichem Aufwand verbunden.* Als verantwortlicher Inbetriebnahme-Koordinator sollte man hier nach eigenem Ermessen entscheiden, ob die Einrichtung eines Syslog-Host zum Zwecke einer höheren Manipulationssicherheit nötig ist und diesen Mehraufwand rechtfertigt.