1 Configuratie van de Apache webserver

We hebben in deze tekst af en toe verwezen naar een 'webserver' en aangehaald dat hij op de juiste manier moet worden geconfigureerd. Er zijn vanzelfsprekend verschillende softwarepakketten op de markt die als webserver dienst kunnen doen, en het is niet onze bedoeling om ze hier allemaal te overlopen.

Om je toch een idee te geven van wat voor mogelijkheden een professionele webserver allemaal biedt, bespreken we hier in het kort de configuratie van *Apache*. Merk op dat we absoluut niet de bedoeling hebben om volledig te zijn in onze beschrijving.

Apache is een *modulaire* server : het programma bestaat uit verschillende onderdelen die naar believen kunnen in- of uitgeschakeld worden. Bij de standaardinstallatie zijn reeds de meest courante modules geactiveerd, maar met behulp van de juiste configuratieopdrachten kunnen modules gemakkelijk worden toegevoegd. Verschillende modules worden standaard bijgeleverd, andere modules kan je van het Internet ophalen en in uiterste nood kan je bovendien zelf modules bijprogrammeren.

1.1 Het programma httpd

Onder UNIX heet de Apache webserver **httpd**. Deze daemon wordt doorgaans opgestart als onderdeel van de gebruikelijke opstartscripts. Het is ook mogelijk om **httpd** in de voorgrond te draaien om op die manier bepaalde configuratieproblemen gemakkelijker op te sporen.

Apache maakt gebruik van verschillende configuratiebestanden[1] :

/etc/httpd/conf/httpd.conf

> Bevat configuratieopdrachten die te maken hebben met de algemene werking van de server.

/etc/httpd/conf.d/.conf*

> Bevat modulespecifieke configuratiebestanden. Bijvoorbeeld ssl.conf voor het gebruik van Secure Socket Layer.

De meeste configuratie-opdrachten zijn lijnen die beginnen met een bepaald sleutelwoord en voor de rest een aantal parameters bevatten. Commentaarlijnen beginnen met een kruis (#).

1.2 Secties

De configuratieopdrachten kunnen ook worden samengebracht in zogenaamde *secties*. Dergelijke secties worden aangegeven met een syntax die enigszins lijkt op die van HTML-tags : eerst komt een sectiehoofding tussen scheve haken (<. . .>), daarna volgen een aantal gewone configuratielijnen en tot slot wordt de sectie afgesloten met een tag van de vorm </. . .>.

Een dergelijke sectie beperkt de geldigheid van de omsloten configuratieopdrachten tot een bepaalde context. We geven een overzicht van de meest gebruikelijke secties :

`<Directory naam>...</Directory>`

> Deze opdracht zorgt ervoor dat de omsloten configuratieopdrachten slechts gelden in de directory met de opgeven naam, of in subdirectories ervan —

[1] De server schrijft ook informatie naar een aantal logbestanden in de directory */etc/httpd/logs*.

tenzij er elders voor een bepaalde subdirectory een afzonderlijke sectie werd gedefinieerd. De *naam* mag ook de jokertekens `*` of `?` bevatten.

`<Location` *URL*`>...</Location>`

Deze sectie heeft een gelijkaardige betekenis als `<Directory>` maar beperkt configuratieopdrachten nu tot een bepaalde URL (en zijn deelURLs) in plaats van een directory op de server. De parameter *URL* wordt genoteerd als een absoluut pad en mag geen protocol of servernaam bevatten.

`<VirtualHost` *IP-adres [:poortnr]*`>...</VirtualHost>`

Hiermee kan je verschillende configuraties opgeven voor verschillende virtuele servernamen[2].

1.3 Basisinstellingen

ServerName *domeinnaam*

Bepaalt de canonische servernaam van deze server. In principe kan de server zijn eigen naam automatisch bepalen, maar de praktijk leert dat het geen slecht idee is om de naam steeds expliciet op te geven.

Listen *poortnummer*

Het TCP-poortnummer waarop de server nar HTTP-aanvragen moet luisteren. Meestal is dit 80, maar ook 8008 en 8080 worden vaak gebruikt — meestal wanneer de server niet door root, maar door een gewone gebruiker werd opgestart, want dan zijn poortnummers kleiner dan 1024 niet bruikbaar.

User *gebruikersnaam*

Geeft de UNIX-gebruiker aan als wie het serverprogramma aanvragen zal behandelen. Deze gebruiker moet dus op zijn minst leestoegang hebben tot de webdirectories en moet in staat zijn om de verschillende CGI-scripts uit

[2]We veronderstellen hier dat verschillende virtuele servernamen ook met verschillende IP-adressen overeenkomen. Sinds HTTP/1.1 laat Apache ook toe om verschillende namen te gebruiken met hetzelfde adres, maar dit vereist nog extra configuratielijnen. Voor meer informatie verwijzen we naar de online documentatie of de website *http://www.apache.org/*.

te voeren. Het is *geen* goed idee om hier de rootgebruiker in te vullen. In de praktijk zal meestal een afzonderlijke gebruiker voor dit doel creëren.

De opdracht `Group` heeft dezelfde betekenis, maar dan voor de UNIX-groep.

DirectoryIndex *naam1 naam2* ...

Geeft de achtereenvolgende bestandsnamen op die de server moet proberen wanneer een URL werd opgevraagd waarvan de naam eindigt met een schuine streep. Bijvoorbeeld :

```
DirectoryIndex index.htm index.html welcome.htm
```

1.4 Omzetten van URL's naar bestandsnamen

Een aantal configuratieopdrachten wordt gebruikt om URL's om te zetten naar bestandsnamen :

DocumentRoot *directorynaam*

Hiermee geef je de basisdirectory aan van waaruit **httpd** bestanden ophaalt die hij naar de client stuurt. De opgegeven *directorynaam* correspondeert met de URL '/'. In het onderstaande voorbeeld gebruiken we twee dergelijke opdrachten om de webdirectories voor twee verschillende virtuele hosts te specifiëren.

```
<VirtualHost 193.190.172.16>
  DocumentRoot /usr/local/WWW/gonzo
  ...
</VirtualHost>

<VirtualHost 192.168.16.15>
  DocumentRoot /usr/local/WWW/inwe
  ...
</VirtualHost>
```

Alias *URL padnaam*

Deze opdracht geeft aan dat alle URL's die beginnen met de opgegeven *URL* moeten worden vertaald naar bestandsnamen die beginnen met de gegeven *padnaam*. In dat geval wordt er niet naar de opgegeven `DocumentRoot`

gekeken. Vaak zal er voor de *padnaam* ook nog een afzonderlijke sectie moeten worden geconfigureerd.

```
DocumentRoot /home/local/WWW
Alias /doc /usr/doc
Alias /local /home/local/html
```

In dit voorbeeld worden alle URL's gezocht in */home/local/WWW*, behalve deze die beginnen met */doc* of */local*.

ScriptAlias *URL padnaam*

Deze opdracht werkt op dezelfde manier als `Alias` maar geeft bovendien aan dat de URL niet met een gewoon bestand overeenkomt, maar met een CGI-script. (Merk op dat je voor scripts ook nog de `ExecCGI` optie nodig hebt — zie §1.5.)

```
ScriptAlias /bin/ /home/local/WWW/bin/
ScriptAlias /cgi-bin/ /home/local/WWW/bin/
```

Een aanvraag voor URL */cgi-bin/cgidump* zal in dit voorbeeld het script */home/local/WWW/bin/cgidump* activeren.

UserDir *patroon*

URL's die beginnen met een tilde (~) verwijzen doorgaans naar bestanden van afzonderlijke gebruikers en niet naar de algemene website van de server. Met de opdracht `UserDir` kan je aangeven in welke directory de corresponderende bestanden moeten worden gezocht.

Onderstaande tabel toont het effect van een aantal verschillende `UserDir`-opdrachten op de URL '~*user1/i3/index.htm*' :

Configuratieopdracht	Padnaam v/h bestand
`UserDir www`	~*user1/www/i3/index.htm*
`UserDir /usr/www/`	*/usr/www/user1/i3/index.htm*
`UserDir /home/*/www`	*/home/user1/www/i3/index.htm*

1.5 Opties

De configuratieopdracht `Options` laat toe om een aantal extra servermogelijkheden (binnen een bepaalde directorysectie) aan te schakelen. Deze opties worden opgegeven als parameter van de opdracht — je kan gerust meerdere parameters tegelijk opgeven. In onderstaande tabel vermelden we enkele van de vele mogelijkheden.

ExecCGI	CGI-scripts in deze directory mogen worden uitgevoerd.
FollowSymLinks	De server mag symbolische links volgen in deze directory.
Indexes	Toon in de plaats een directorylisting wanneer geen enkel bestand uit de opgegeven `DirectoryIndex` blijkt te bestaan, bij een URL die eindigt op '/'.

De `ScriptAlias`-opdracht moet je steeds gebruiken in combinatie met de `ExecCGI`-optie zodat de aangeduide CGI-scripts ook daadwerkelijk zouden kunnen worden uitgevoerd.

```
ScriptAlias /cgi-bin/ /home/local/WWW/bin/
<Directory /home/local/WWW/bin>
Options ExecCGI
</Directory>
```

1.6 Toegangscontrole

Apache biedt heel wat mogelijkheden om toegang te regelen tot bepaalde URL's of directories. We lichten hier slechts een klein tipje van de sluier op.

AuthType Basic

Geeft aan welk type authenticatie onze server zal gebruiken. Het alternatief is `AuthType Digest`.

De volgende opdrachten bevinden zich in een `<Directory>`-sectie en beperken de toegang tot de corresponderende directory.

AuthName *rijk*

Geeft de naam aan van het rijk (realm) dat voor authenticatie zal worden gebruikt (voor deze directory).

AuthUserFile *padnaam*

Geeft de padnaam op van een bestand dat gebruikers en wachtwoorden bevat die bij authenticatie moeten worden gebruikt. Elke lijn in dit bestand bestaat uit een gebruikersnaam een dubbele punt en een geencrypteerd wachtwoord.

Het Apache-pakket bevat een programma **htpasswd** dat je kan gebruiken om dergelijke wachtwoordbestanden te beheren.

require user *gebruiker1 gebruiker2* ...

Geeft aan welke gebruikers er toegang hebben tot deze directory. De gebruikers worden opgezocht in het wachtwoordbestand dat met `AuthUserFile` werd ingesteld en de corresponderende wachtwoorden worden vergeleken met deze die door de HTTP-client worden doorgegeven.

require valid-user

Bepaalt dat elke gebruiker toegang krijgt tot deze directory, op voorwaarde dat hij met succes op het corresponderende rijk is ingelogd.

Deze vijf configuratieopdrachten komen meestal tezamen voor :

```
<Directory /usr/local/WWW/intranet>
AuthType Basic
AuthName "Triple Eye Intranet"
AuthUserFile /home/local/intranet/passwd
require user iii
</Directory>
```

Toegang tot bepaalde directories kan ook worden beperkt aan de hand van IP-adressen of domeinnamen. Hiertoe dienen de opdrachten `allow` en `deny`. Een aantal voorbeelden :

```
<Directory /usr/local/WWW/intranet>
order deny,allow
deny from all
allow from .hogent.be
</Directory>
```

(Geef enkel toegang aan clients die tot het domein *hogent.be* behoren.)

```
<Directory /usr/local/WWW/almostpublic>
order allow,deny
allow from all
deny from 192.168.16
</Directory>
```

(Elke client krijgt toegang, behalve deze van het subnet 192.168.16.*)

order deny,allow

Toegang wordt geweigerd aan alle toestellen die aan een `deny`-opdracht voldoen, tenzij ze zich ook in een `allow`-opdracht bevinden. Toestellen die niet in de `deny`-opdracht worden vermeld, krijgen toegang.

order allow,deny

Toegang wordt geweigerd aan alle toestellen die niet in de `allow`-opdracht worden vermeld. Toestellen die aan de `allow`-opdracht voldoen maar ook in de `deny`-opdracht voorkomen, krijgen ook geen toegang.

Wanneer tegelijkertijd op dezelfde directory authenticatie op naam en toegangsbeperking volgens clientmachine wordt toegepast, bepaalt de `Satisfy`-opdracht wat het resultaat zal zijn : '`Satisfy all`' betekent dat beide voorwaarden moeten voldaan zijn om toegang te krijgen, terwijl met '`Satisfy any`' één van beide voorwaarden volstaat.

Het volgende configuratievoorbeeld bepaalt dat het 'Triple Eye Intranet' vanzelf toegankelijk is voor elke machine met één van de opgegeven netwerkadressen, of ook van buitenaf als men tenminste het wachtwoord van de gebruiker 'iii' kent.

```
<Directory /usr/local/WWW/intranet>
Satisfy any

AuthType Basic
AuthName "Triple Eye Intranet"
AuthUserFile /home/local/intranet/passwd
require user iii

order deny,allow
deny from all
allow from 192.168.12
allow from 193.190.172
</Directory>
```