4.0 Setup Zero Footprint Apache Web Server on Linux v0.3

This document covers zero footprint compilation and installation of the Apache HTTP Server. It also covers the basic configuration settings to allow the compilation of a server corresponding to specific requirements. Also best practice to create secure web server are provided , which can be applied based on application and security need.

Create Zero FootPrint Apache Package

Create No Login User

Create apacheaem no login user without home directory. This user is solely created to run apache services

```
sudo adduser apacheaem --shell=/bin/false --no-create-home
```

Install C Compiler

Compiling Apache Server

```
sudo apt-get update
sudo apt-get install build-essential
```

Download Apache Package

```
cd /home/bnsadm/
wget https://www.apache.org/dist/httpd/httpd-2.2.32.tar.bz2
tar -xvf httpd-2.2.32.tar.bz2
```

Configure apache for compilation

```
cd /home/bnsadm/httpd-2.2.32
./configure --prefix=/home/bnsadm/apache2 --enable-mods-shared=few
--enable-module-rewrite
```

--prefix=/home/bnsadm/apache2 will set a root directory where apache will get installed
--enable-mods-shared=few option will compile basic set of modules.

There are several configurations options which can be set based on need, refer to below site to get more details on ./configure setting options.

http://httpd.apache.org/docs/2.4/programs/configure.html#installationdirectories

Prepare Build and Compile Apache

execute below command to prepare build and compile apache under /home/bnsadm/apache2 folder as defined by --prefix option in above step.

```
make
make install
```

Make Config Changes

```
cd /home/bnsadm/apache2/conf/
cp -p httpd.conf httpd.conf_orig
sed -e 's/User daemon/User apacheaem/g; s/Group daemon/Group apacheaem/g;
s/#ServerName www.example.com:80/ServerName localhost/g'  httpd.conf >>
httpd.conf_1
mv httpd.conf_1 httpd.conf
```

Above command will make apache services to run with apacheaem no login user. It will also change ServerName to localhost.
Any other config changes related to apache dispatcher can be added in sed command separated with `;`.

Start and test Apache

```
sudo /home/bnsadm/apache2/bin/apachectl start
curl http://localhost:80
```

The reason why apache needs to run as root initially is because by default apache will needs to bind itself to port 80/TCP. Anything that runs below port 1024 needs to be root. However this will only be for the parent process, the subsequent child processes will run as apacheaem nologin user.

Package Apache for Zero FootPrint Deployment on other machines

...

The tar command used here will not work properly.

```
cd /home/bnsadm/
tar -cvzf apache_zerofs.tar.gz apache2
```

Deploy and start Zero FootPrint Apache Server on other machine

Copy apace_zerofs.tar.gz file to another machine and execute below command to setup and start apache server.

```
tar -xvf apache_zerofs.tar.gz
cd apache2
sudo bin/apachectl start
curl http://localhost:80
```

Please note that apacheaem no login user will have to be created on this machine.

Best practices for secure web server

You may follow these best practices to secure Apache Web Server on UNIX / Linux machine

Disable unnecessary modules

If you are planning to install apache, you should disable the following modules. If you do ./configure –help, you'll see all available modules that you can disable/enable.

- userdir – Mapping of requests to user-specific directories. i.e ~username in URL will get translated to a directory in the server
- autoindex – Displays directory listing when no index.html file is present
- status – Displays server stats
- env – Clearing/setting of ENV vars
- setenvif – Placing ENV vars on headers
- cgi – CGI scripts
- actions – Action triggering on requests
- negotiation – Content negotiation
- alias – Mapping of requests to different filesystem parts
- include – Server Side Includes
- filter – Smart filtering of request
- version – Handling version information in config files using IfVersion
- as-is – as-is filetypes

Disable all of the above modules as shown below when you do ./configure

```
./configure \
--enable-ssl \
--enable-so \
--disable-userdir \
--disable-autoindex \
--disable-status \
--disable-env \
--disable-setenvif \
--disable-cgi \
--disable-actions \
--disable-negotiation \
--disable-alias \
--disable-include \
--disable-filter \
--disable-version \
--disable-asis
```

After the installation, when you do **httpd -l**, you'll see all installed modules.

```
# /home/bnsadm/apache2/bin/httpd -l
Compiled in modules:
  core.c
  mod_authn_file.c
  mod_authn_default.c
  mod_authz_host.c
  mod_authz_groupfile.c
  mod_authz_user.c
  mod_authz_default.c
  mod_auth_basic.c
  mod_log_config.c
  mod_ssl.c
  prefork.c
  http_core.c
  mod_mime.c
  mod_dir.c
  mod_so.c
```

In this example, we have the following apache modules installed.

- core.c – Apache core module
- mod_auth* – For various authentication modules
- mod_log_config.c – Log client request. provides additional log flexibilities.
- mod_ssl.c – For SSL
- prefork.c – For MPM (Multi-Processing Module) module
- httpd_core.c – Apache core module
- mod_mime.c – For setting document MIME types
- mod_dir.c – For trailing slash redirect on directory paths. if you specify url/test/, it goes to url/test/index.html
- mod_so.c – For loading modules during start or restart

Run Apache as separate user and group

By default, apache might run as nobody or daemon. It is good to run apache in its own non-privileged account as in example we have configured to run it as apacheaem.

Create apache group and user.

```
groupadd apacheaem
useradd -d /home/bnsadm/apache2/htdocs -g apache -s /bin/false apacheaem
```

Modify the httpd.conf, and set User and Group appropriately.

```
# vi httpd.conf
User apacheaem
Group apacheaem
```

After this, if you restart apache, and do ps -ef, you'll see that the apache is running as "apacheaem" (Except the 1st httpd process, which will always run as root).

```
# ps -ef | grep -i http | awk '{print $1}'
```

Restrict access to root directory (Use Allow and Deny)

Secure the root directory by setting the following in the **httpd.conf**

```
<Directory />
    Options None
    Order deny,allow
    Deny from all
</Directory>
```

In the above:

- **Options None** – Set this to None, which will not enable any optional extra features.
- **Order deny,allow** – This is the order in which the “Deny” and “Allow” directivites should be processed. This processes the “deny” first and “allow” next.
- **Deny from all** – This denies request from everybody to the root directory. There is no Allow directive for the root directory. So, nobody can access it.

Set appropriate permissions for conf and bin directory

bin and conf directory should be viewed only by authorized users. It is good idea to create a group, and add all users who are allowed to view/modify the apache configuration files to this group.

Let us call this group: apacheadmin

Create the group.

```
groupadd apacheadmin
```

Allow access to bin directory for this group.

```
chown -R root:apacheadmin /home/bnsadm/apache2/bin
chmod -R 770 /home/bnsadm/apache2/bin
```

Allow access to conf directory for this group.

```
chown -R root:apacheadmin /home/bnsadm/apache2/conf
chmod -R 770 /home/bnsadm/apache2/conf
```

Add appropriate members to this group. In this example, both ramesh and john are part of apacheadmin

```
# vi /etc/group
apacheadmin:x:1121:rinku,tin, adam
```

Disable Directory Browsing

If you don’t do this, users will be able to see all the files (and directories) under your root (or any sub-directory).

For example, if they go to http://{your-ip}/images/ and if you don’t have an index.html under images, they’ll see all the image files (and the sub-directories) listed in the browser (just like a ls -1 output). From here, they can click on the individual image file to view it, or click on a sub-directory to see its content.

To disable directory browsing, you can either set the value of **Options directive to “None” or “-Indexes”**. A – in front of the option name will remove it from the current list of options enforced for that directory.

Indexes will display a list of available files and sub-directories inside a directory in the browser (only when no index.html is present inside that folder). So, Indexes should not be allowed.

```
<Directory />
  Options None
  Order allow,deny
  Allow from all
</Directory>

(or)

<Directory />
  Options -Indexes
  Order allow,deny
  Allow from all
</Directory>
```

Don’t allow .htaccess

Using .htaccess file inside a specific sub-directory under the htdocs (or anywhere ouside), users can overwrite the default apache directives. On certain situations, this is not good, and should be avoided. You should disable this feature.

You should not allow users to use the .htaccess file and override apache directives. To do this, set “**AllowOverride None**” in the root directory.

```
<Directory />
  Options None
  AllowOverride None
  Order allow,deny
  Allow from all
</Directory>
```

Disable other Options

Following are the available values for Options directive:

- **Options All** – All options are enabled (except MultiViews). If you don’t specify Options directive, this is the default value.
- **Options ExecCGI** – Execute CGI scripts (uses mod_cgi)
- **Options FollowSymLinks** – If you have symbolic links in this directory, it will be followed.
- **Options Includes** – Allow server side includes (uses mod_include)
- **Options IncludesNOEXEC** – Allow server side includes without the ability to execute a command or cgi.
- **Options Indexes** – Disable directory listing
- **Options MultiViews** – Allow content negotiated multiviews (uses mod_negotiation)
- **Options SymLinksIfOwnerMatch** – Similar to FollowSymLinks. But, this will follow only when the owner is same between the link and the original directory to which it is linked.

Never specify ‘Options All’. Always specify one (or more) of the options mentioned above. You can combine multiple options in one line as shown below.

```
Options Includes FollowSymLinks
```

The + and – in front of an option value is helpful when you have nested direcotires, and would like to overwrite an option from the parent Directory directive. In this example, for /site directory, it has both Includes and Indexes:

```
<Directory /site>
  Options Includes Indexes
  AllowOverride None
  Order allow,deny
  Allow from all
</Directory>
```

For /site/en directory, if you need Only Indexes from /site (And not the Includes), and if you want to FollowSymLinks only to this directory, do the following.

```
<Directory /site/en>
  Options -Includes +FollowSymLink
  AllowOverride None
  Order allow,deny
  Allow from all
</Directory>
```

- /site will have Includes and Indexes
- /site/en will have Indexes and FollowSymLink

Remove unwanted DSO modules

If you have loaded any dynamic shared object modules to the apache, they'll be present inside the httpd.conf under "LoadModule" directive.Please note that the statically compiled apache modules will not be listed as "LoadModule" directive.Comment out any unwanted **"Load Modules"** in the httpd.conf

```
grep LoadModule /home/bnsadm/apache2/conf/httpd.conf
```

Restrict access to a specific network (or ip-address)

If you want your site to be viewed only by a specific ip-address or network, do the following: To allow a specific network to access your site, give the network address in the Allow directive.

```
<Directory /site>
  Options None
  AllowOverride None
  Order deny,allow
  Deny from all
  Allow from 10.10.0.0/24
</Directory>
```

To allow a specific ip-address to access your site, give the ip-address in the Allow directive.

```
<Directory /site>
  Options None
  AllowOverride None
  Order deny,allow
  Deny from all
  Allow from 10.10.1.21
</Directory>
```

Don't display or send Apache version (Set ServerTokens)

By default, the server HTTP response header will contains apache and php version. Something similar to the following. This is harmful, as we don't want an attacker to know about the specific version number.

```
Server: Apache/2.2.17 (Unix) PHP/5.3.5
```

To avoid this, set the ServerTokens to Prod in httpd.conf. This will display "Server: Apache" without any version information.

```
# vi httpd.conf
ServerTokens Prod
```

Following are possible ServerTokens values:

- **ServerTokens Prod** displays "Server: Apache"
- **ServerTokens Major** displays "Server: Apache/2"
- **ServerTokens Minor** displays "Server: Apache/2.2"
- **ServerTokens Min** displays "Server: Apache/2.2.17"
- **ServerTokens OS** displays "Server: Apache/2.2.17 (Unix)"
- **ServerTokens Full** displays "Server: Apache/2.2.17 (Unix) PHP/5.3.5" (If you don't specify any ServerTokens value, this is the default)