



Apache

DOCUMENTATION MANUAL FOR APACHE HTTP SERVER

ADRIANO, JOHN CARLO

BANGIACAN, MARIA STEPHANIE KRISTINA

LICUDO, MARY ANN

OLIVAS, MARVIN

RAMOS, TATUM EIFFEL DODGE

ZHANG, JASPER ANTHONY

Table of Contents

Title Page	i
Table of Contents	ii
Ubuntu V1irtual Host	1
Enable Compression	5
Content Caching	6
Content Negotiation	7
Accept	7
Accept-Language	8
Authentication and Authorization of HTTP Requests	10
Server – Side Includes (SSI)	12
XBitHack directive	12
Filename extension-based	14
HTTPS Redirect with Self – Signed Certificate	15

UBUNTU VIRTUAL HOST

1. After logging in to the server, create a directory where you can place your files in the path, `/var/www` with the `mkdir` command and `-p` option

Ours, inside the folder `group3a.com` we have another folder `public_html`

```
webtech@webtech2018:/var/www$ sudo mkdir -p /var/www/group3a.com/public_html
```

I'll create two more directories for `group3b` and `group3c`

```
webtech@webtech2018:~$ sudo mkdir -p /var/www/group3b.com/public_html
webtech@webtech2018:~$ sudo mkdir -p /var/www/group3c.com/public_html
webtech@webtech2018:~$
```

2. Issue the command, `chown` to change the ownership of the files and directories in a Linux filesystem. Also affix the proper change mode values. To do so,

```
webtech@webtech2018:/var/www$ sudo mkdir -p /var/www/group3a.com/public_html
webtech@webtech2018:/var/www$ sudo chown -R $USER:$USER /var/www/group3a.com/public_html
webtech@webtech2018:/var/www$ sudo chmod -R 755 /var/www
webtech@webtech2018:/var/www$
```

3. Next, go inside your `public_html` folder and create your html file. In some case where you have your files in an online repository *clone* your repository inside the `public_html` folder.

We have our files in github and used the `git clone` command as shown in the highlighted command below.

To check if you successfully clone the repository issue the list command, `ls`

```
webtech@webtech2018:/var/www$ cd /var/www/group3a.com/public_html
webtech@webtech2018:/var/www/group3a.com/public_html$ git clone https://github.com/2160563/repo2
Cloning into 'repo2' ...
remote: Counting objects: 27, done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 27 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (27/27), done.
Checking connectivity... done.
webtech@webtech2018:/var/www/group3a.com/public_html$
```

Figure 1
Cloning

Same command is to be issued to clone your files in your repository for the case of `group3b` and `group3c` folders. Make sure you type in the correct address or path in cloning your repositories.

4. Then, configure your virtual host file by entering `cd /etc/apache2` you will be directed to the said directory. Optionally, you can always issue the `ls` command to check files or folders that are present in your present working directory.

```
webtech@webtech2018:/etc/apache2$ ls
apache2.conf  conf-enabled  magic          mods-enabled  sites-available
conf-available  envvars      mods-available  ports.conf    sites-enabled
webtech@webtech2018:/etc/apache2$ _
```

Next, type `cd sites-available`. Inside *sites-available* is the default configuration file, named *000-default.conf* and *default-ssl.conf*

But we are not going to use the default one instead provides a virtual host configuration file. We issue the command `cp`.

Type in the command as shown below, the command allows you to copy the default configuration file to a new source file which you named, *group3a.com.conf*

```
webtech@webtech2018:/etc/apache2$ cd sites-available
webtech@webtech2018:/etc/apache2/sites-available$ ls
000-default.conf  btnhd.com.conf  default-ssl.conf  humanrights.com.conf
webtech@webtech2018:/etc/apache2/sites-available$ sudo cp /etc/apache2/sites-available/000-default.c
nf /etc/apache2/sites-available/group3a.com.conf
[sudo] password for webtech:
webtech@webtech2018:/etc/apache2/sites-available$ ls
000-default.conf  btnhd.com.conf  default-ssl.conf  group3a.com.conf  humanrights.com.conf
webtech@webtech2018:/etc/apache2/sites-available$
```

Figure 2
Copy

Make sure the newly created configuration file is in the list as you issue the `ls` command.

You are expected to do the same thing for *group3b* and *group3c*, but make sure to provide a new configuration file name (ie. *group3b.com.conf*, *group3c.com.conf*)

- 5. You now edit the three separate configuration file for *group3a.com.conf*, *group3b.com.conf* and *group3c.com.conf*,

Type `sudo nano group3a.com.conf` and add the command below

```
GNU nano 2.5.3      File: group3a.com.conf

<VirtualHost *:80>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.

    ServerName www.group3a.org
    ServerAlias group3a.org
    ServerAdmin info@group3a.com
    DocumentRoot /var/www/group3a.com/public_html/repo2

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn
```

Figure 3
Configuring Virtual Host

Type `sudo nano group3b.com.conf`, add and modify the contents

Type `sudo nano group3c.com.conf`, add and modify the contents

Edit the following

ServerName, usually referred as domain name mapped to the ip address

ServerAlias, alternate name accepted by the server

ServerAdmin, it sets the contact address for the server to return any error messages to the client

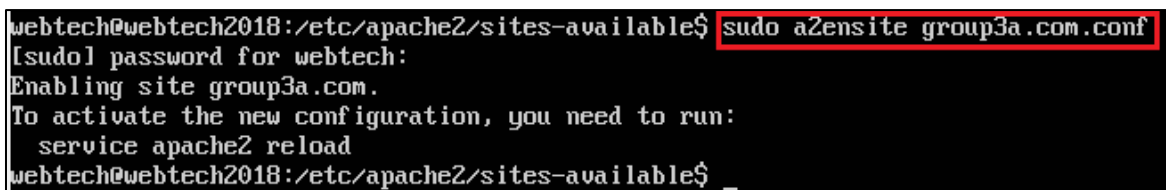
DocumentRoot, the specific path of your document which will be accessed by the virtual host

Make sure you entered the right path of your html files and stylesheets in the *DocumentRoot*, for this will fetch the files in the said path.

Save your file.

6. The next step is to enable your virtual host file.

Use the **a2ensite** command

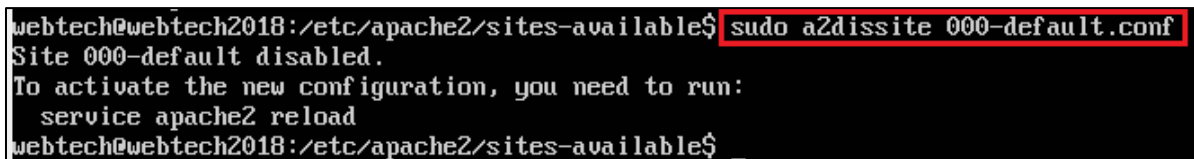


```
webtech@webtech2018:/etc/apache2/sites-available$ sudo a2ensite group3a.com.conf
[sudo] password for webtech:
Enabling site group3a.com.
To activate the new configuration, you need to run:
  service apache2 reload
webtech@webtech2018:/etc/apache2/sites-available$ _
```

Figure 4
Enabling Site

Enable the site also for *group3b.com.conf* and *group3c.com.conf*

And disable the default website, using the **a2dissite** command

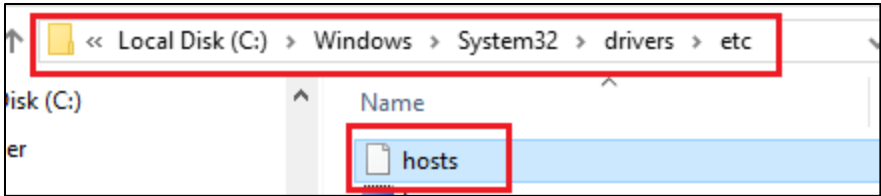


```
webtech@webtech2018:/etc/apache2/sites-available$ sudo a2dissite 000-default.conf
Site 000-default disabled.
To activate the new configuration, you need to run:
  service apache2 reload
webtech@webtech2018:/etc/apache2/sites-available$ _
```

Figure 5
Disabling Site

Optionally, you can go to sites-enabled folder to see the list of configuration file that is/are enabled.

7. Restart your apache web server. Issue the command, **sudo service apache2 restart**
8. You've successfully host your website locally via the apache web server. To test, go to the path specified below:



And edit your hosts file; include the IP address taken from your host machine and the domain name of your website (known to be the *ServerName* in your configuration file). Remember to save your hosts file.

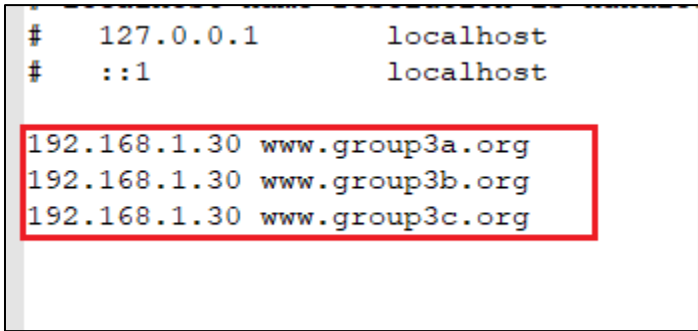


Figure 6
Entry in hosts file

Open a web browser and type in the domain name, www.group3a.org , www.group3b.org and www.group3c.org in the address bar.

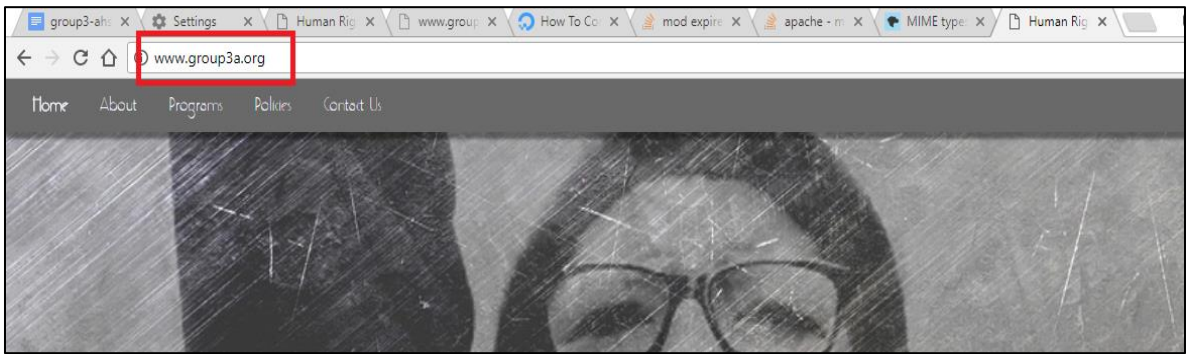


Figure 7
Human Rights Website



Figure 8
Inclusion Website

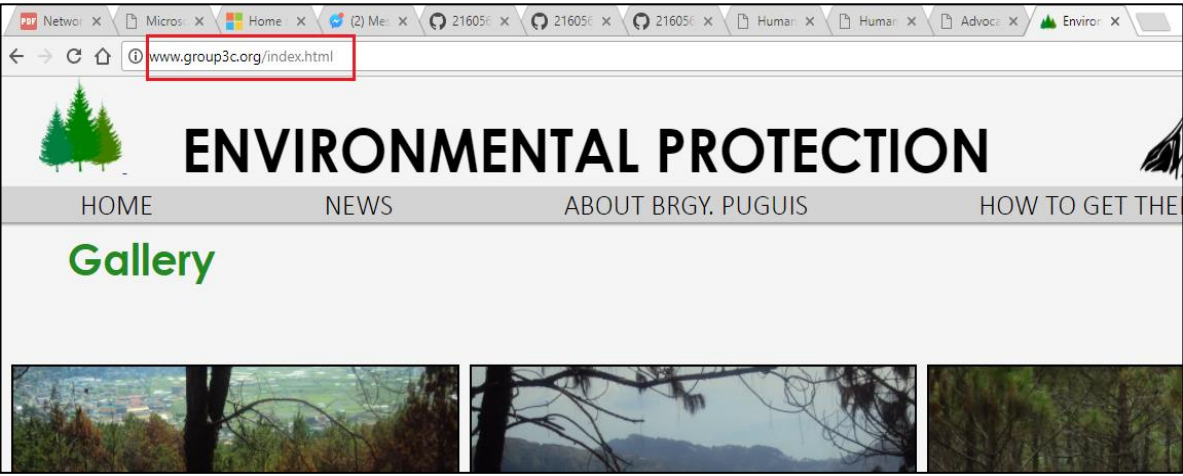


Figure 9
Environmental Protection Website

ENABLE COMPRESSION

This shows a step by step procedure to enable compression of contents such as *html* and *css* files. Assuming you were able to host your websites in the apache web server, you can do the following steps:

Go to the directory *apache2*, type the command `cd /etc/apache2` and issue `ls` command to check the list of the directory.

```
webtech@webtech2018:~$ cd /etc/apache2
webtech@webtech2018:/etc/apache2$ ls
apache2.conf  conf-enabled  magic          mods-enabled  sites-available
conf-available  envvars      mods-available  ports.conf    sites-enabled
webtech@webtech2018:/etc/apache2$
```

Type `cd sites-enabled` and issue the command `sudo nano group3a.com.conf`. Configure the virtual host file in such a way that it will compress the *html* and *css* file.

To do so, add the the configuration command in the virtual host file as seen below.

```
GNU nano 2.5.3      File: group3a.com.conf
<VirtualHost *:80>

    ServerName www.group3a.org
    ServerAlias group3a.org
    ServerAdmin info@group3a.com
    DocumentRoot /var/www/group3a.com/public_html/repo2

    <IfModule mod_deflate.c>
        <IfModule mod_filter.c>
            AddOutputFilterByType DEFLATE text/html
            AddOutputFilterByType DEFLATE text/css
        </IfModule>
    </IfModule>
```

Figure 10
Compression

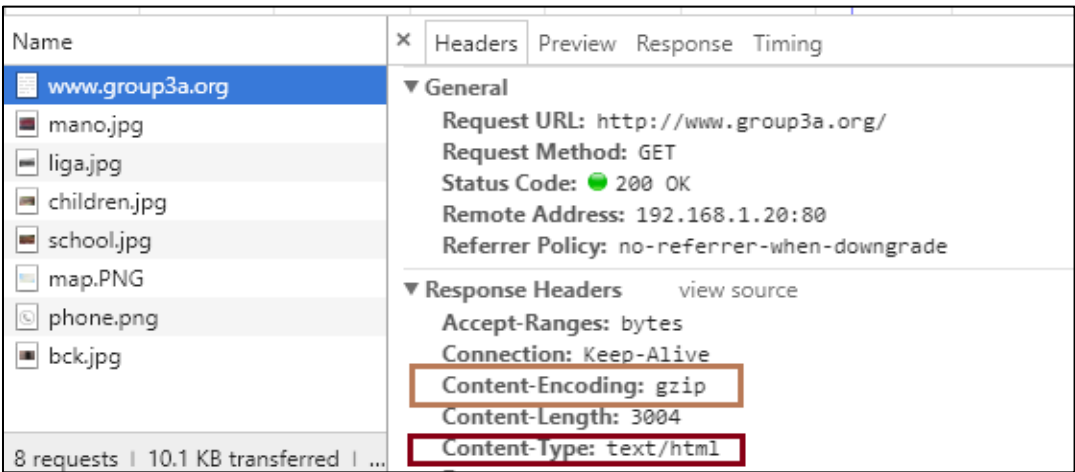
Save your file.

Restart your apache web server. Issue the command, **sudo service apache2 restart**

One way to check if your html and css files are rendered in *gzip* format is to browse the website in a browser and *click* the icon



and go to More Tools > Developer Tools > Refresh the webpage



Notice the Content – Type: text/html and the Content – Encoding: *gzip* in the Response Header

CONTENT CACHING

The procedures enables the clients to cache PNG, JPG and GIF files for up to 24 hours from the time they are accessed.

First enable the module for caching; type **sudo a2enmod expires** (*in case it is not yet enabled*)

Assuming you are in this path : */etc/apache2/sites-available*

Type **sudo nano group3a.com.conf** and add the following commands to enable caching for images with the span of 1 day

```
<IfModule mod_expires.c>
    ExpiresActive on
    <filesMatch ".(gif|jpg|png)$">
        ExpiresDefault "access plus 1 day"
        CacheHeader on
        CacheMaxExpire 86400
    </filesMatch>
</IfModule>

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
```

Figure 11
Caching

Save the file.

Check for syntax errors, **sudo apachectl configtest**

Restart your apache server.

CONTENT NEGOTIATION

The following steps enable clients to negotiate with the servers, using the HTTP Accept header and the HTTP Accept – language header.

1. Create a virtual host that is mapped to the domain names *webtech1.negotiate.org*.

To do so,

```
sudo mkdir -p /var/www/webtech1
sudo mkdir -p /var/www/webtech1/negotiation
sudo chown -R $USER:$USER /var/www/webtech1/negotiation
sudo chmod -R 755 /var/www
sudo cp /etc/apache2/sites-available/000-default.conf
/etc/apache2/sites-available/webtech1.conf
```

2. Edit the configuration file of the virtual host

Type **sudo nano /etc/apache2/sites-available/webtech1.conf**

Add the following,

A screenshot of the GNU nano 2.5.3 text editor. The title bar shows 'File: /etc/apache2/sites-available/webtech1.conf'. The editor content shows a VirtualHost block for *:80. A red rectangular box highlights the following configuration lines: `ServerName webtech1.negotiate.org`, `ServerAdmin webtec@webtech1.negotiate.org`, and `DocumentRoot /var/www/webtech1/negotiation`.

```
GNU nano 2.5.3 File: /etc/apache2/sites-available/webtech1.conf
<VirtualHost *:80>
-
  ServerName webtech1.negotiate.org
  ServerAdmin webtec@webtech1.negotiate.org
  DocumentRoot /var/www/webtech1/negotiation
```

Save the file.

ACCEPT

- a. In your folder negotiation create two dummy files:

sudo nano content.html

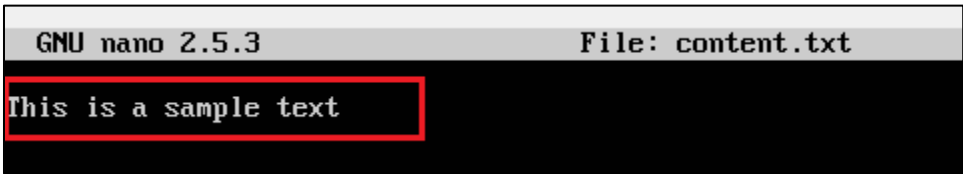
Edit the file and save

A screenshot of the GNU nano 2.5.3 text editor. The title bar shows 'File: content.html'. The editor content shows the following HTML code: `<html>`, `<p> HTML </p>`, and `</html>`. A red rectangular box highlights these three lines.

```
GNU nano 2.5.3 File: content.html
<html>
<p> HTML </p>
</html>
```

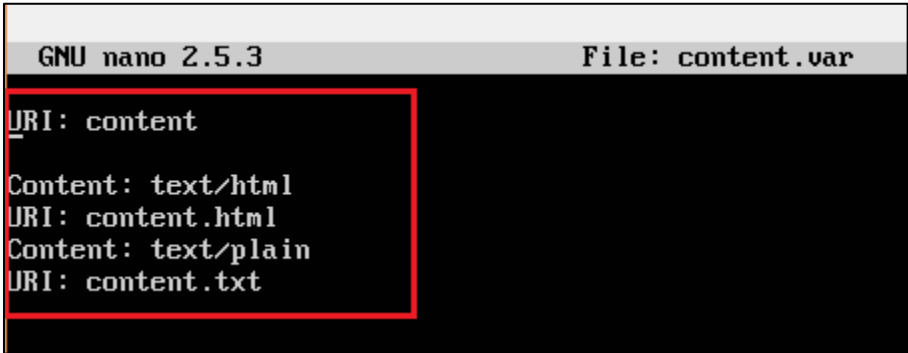
sudo nano content.txt

Again, edit and save



```
GNU nano 2.5.3 File: content.txt
This is a sample text
```

- b. Create a var file named *content.var*. Type `sudo nano content.var`
Inside *content.var* type the following



```
GNU nano 2.5.3 File: content.var
URI: content
Content: text/html
URI: content.html
Content: text/plain
URI: content.txt
```

Figure 12
Content var file

Save the file

ACCEPT – LANGUAGE

- a. Create another two dummy files inside negotiation folder and do the commands
`sudo nano language.html.en`
Edit the file and save



```
GNU nano 2.5.3 File: language.html.en
<html>
  <head>
    <title>English </title>
  </head>
  <body>
    <p> Language English </p>
  </body>
</html>
English
```

`sudo nano language.html.fil`
Again, edit the file and save

```
GNU nano 2.5.3 File: language.html.fil
<html>
  <head><title> Filipino </title> </head>
  <body>
    <p> Filipino Language </p>
  </body>
</html>
Filipino
```

- b. Create a var file named *language.var*.
Type `sudo nano language.var`
Edit the file and save
Inside *language.var* type the following

```
GNU nano 2.5.3 File: language.var
URI: language
Content: text/html
URI: language.html.en
Content: text/html
URI: language.html.fil
```

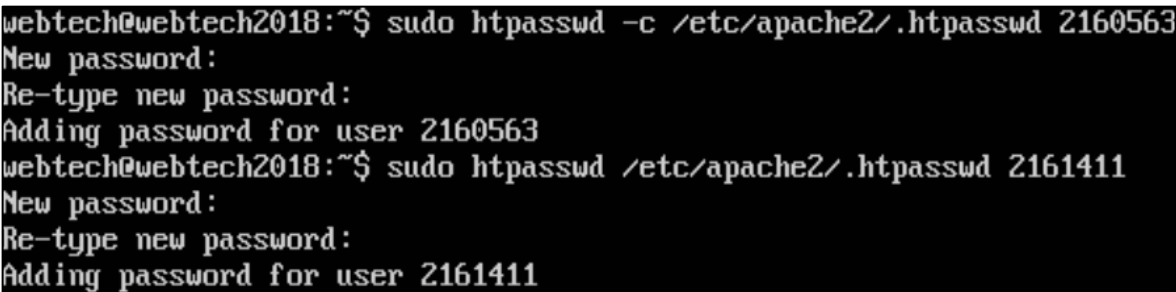
Figure 13
Language var file

- 3. Open the configuration file for the virtual host of webtech1.negotiate.org
Type `sudo nano /etc/apache2/sites-available/webtech1.conf`

<Directory /var/www/webtech1/negotiation>
AddHandler type-map .var
AddLanguage en .en
AddLanguage fil .fil
LangaugePriority en fil
Options +Multiviews
</Directory>
- 4. Enable module, type `sudo a2enmod negotiation`
Enable websites virtual host, type `sudo a2ensite webtech1.conf`
- 5. Reload the web server.

AUTHENTICATION AND AUTHORIZATION OF HTTP REQUESTS

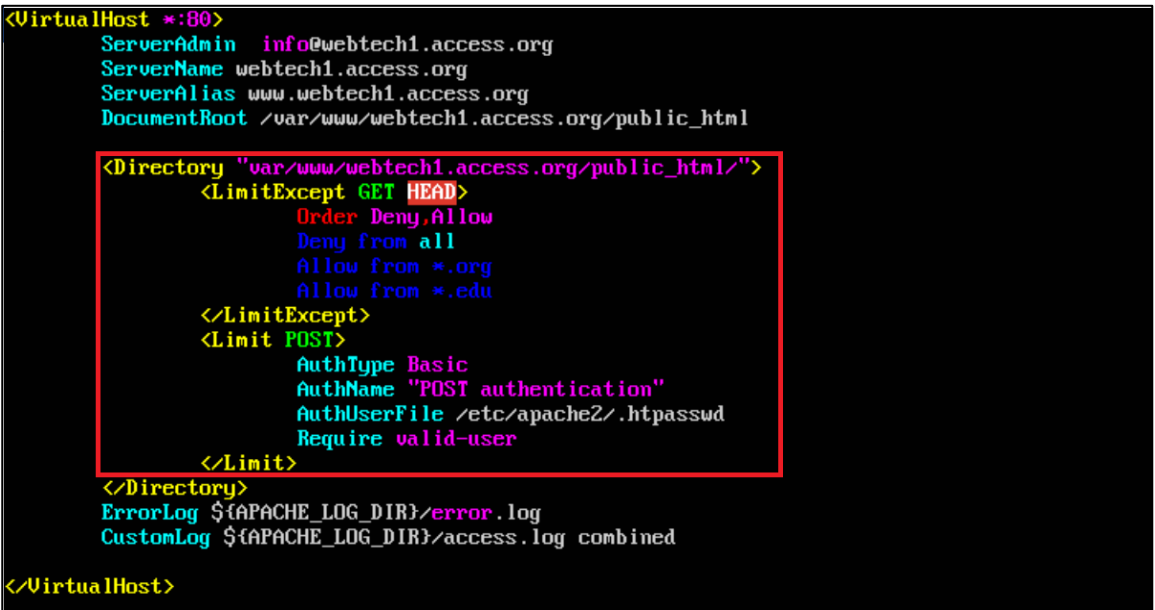
1. Enable the following mods by using `sudo a2enmod <name of the module>`
 - a. `authz_host`
 - b. `authz_user`
 - c. `authz_core`
 - d. `auth_basic`
 - e. `authn_file`
2. Reload apache by using the command `sudo service apache2 reload`
3. Add accounts that are able to use specific HTTP requests.
4. Use the command `sudo htpasswd -c /etc/apache2/.htpasswd <username>` for the first time you add an account `sudo htpasswd /etc/apache2/.htpasswd <username>` to more accounts for then you will be asked to provide a password. See Figure 14



```
webtech@webtech2018:~$ sudo htpasswd -c /etc/apache2/.htpasswd 2160563
New password:
Re-type new password:
Adding password for user 2160563
webtech@webtech2018:~$ sudo htpasswd /etc/apache2/.htpasswd 2161411
New password:
Re-type new password:
Adding password for user 2161411
```

Figure 14
Creating an account in htpasswd file

5. Go to the `/etc/apache2/sites-available` directory.
6. Go to the configuration file of your website. Use the `vi` command for the operating system to allow you to edit the file. For example `sudo vi webtech.access.org.conf`
7. Type the following. See Figure 15



```
<VirtualHost *:80>
    ServerAdmin info@webtech1.access.org
    ServerName webtech1.access.org
    ServerAlias www.webtech1.access.org
    DocumentRoot /var/www/webtech1.access.org/public_html

    <Directory "/var/www/webtech1.access.org/public_html/">
        <LimitExcept GET HEAD>
            Order Deny,Allow
            Deny from all
            Allow from *.org
            Allow from *.edu
        </LimitExcept>
        <Limit POST>
            AuthType Basic
            AuthName "POST authentication"
            AuthUserFile /etc/apache2/.htpasswd
            Require valid-user
        </Limit>
    </Directory>
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Figure 15
Configuration for authentication and authorization

- 8. Save and exit.
- 9. Lastly type `sudo service apache2 restart` and press enter

This is how it looks like if you are asked for authentication to access the website (see figure 7).

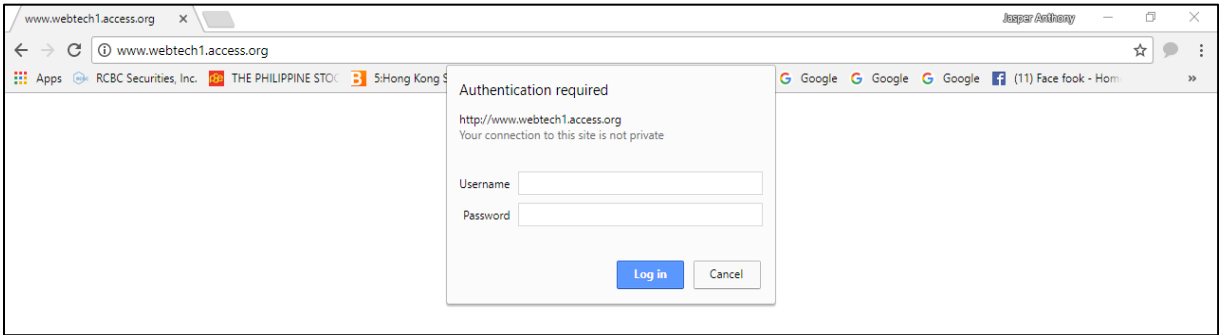


Figure 16
Authentication and Authorization

This is how it looks like if you are forbidden from accessing the website (see figure 8).



Figure 17
Forbidden from accessing the website

This is how it looks like if you are unauthorized from accessing the website (see figure 9).

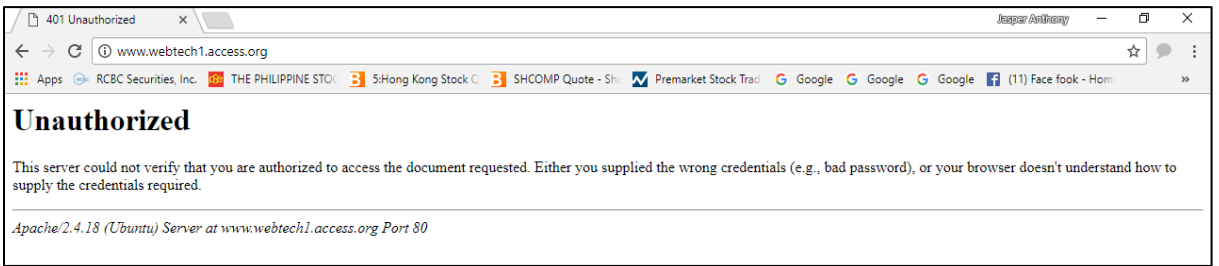


Figure 18
Unauthorized from accessing the website

SERVER-SIDE INCLUDES (SSI)

It is used to include the contents of one or more files to a website inside the server. This tool can also be used for removing repetitive information. Instead of encoding the same things again, you can create a different file that contains that information you want to add then just call or include that file into the HTML.

Note: In order for the server to know that an HTML file is SSI enabled, the file extension should be changed to a **.shtml** one. Or, the execution bit of the file should be set (for the *XBitHack* directive).

XBitHack directive

This method does **NOT** require the file name extension to be set to .shtml.

1. Create a directory, then create the files that you would be needing inside that directory. In our case, we created a header.txt, footer.txt and the index.html.

```
<h1 style="color: red;"> HEADER from header.txt </h1>
```

```
<h6 style="color: blue;"> FOOTER from footer.txt </h6>
```

```
<?DOCTYPE html>
<html lang="en">
  <head>
    <title> SSI </title>
  </head>
  <body>
    <!--#include virtual="header.txt"-->
    <p> Body of index.html XBitHack approach </p>
    <!--#include virtual="footer.txt"-->
  </body>
</html>
```

Figure 19
Html content

The `<!--#include virtual="header.txt"-->` is essential because it is the command that calls the header.txt to the html file. Same goes for the footer.txt.

2. `sudo chmod +x index.html`
this command is used to allow the file to be executable
3. `sudo a2enmod include`
this enables the include module of apache2
4. `sudo service apache2 reload`
reloads the apache web server to cope with the new configuration settings

- 5. `sudo vi /etc/apache2/sites-available/ssi1.conf`
creates a blank configuration file using the `vi` text-editor

Input the commands shown in the figure below for the configuration

Note: Lines beginning with a hash sign (#) are considered as comments

```
# VirtualHost configuration file for enabling
# SSI using the XBitHack directive

<VirtualHost *:80>
    ServerName webtech1.ssi.org
    DocumentRoot /var/www/ssi1

    <Directory /var/www/ssi1>
        Options +Indexes +IncludesNOEXEC +FollowSymLinks
        AllowOverride all
        XBitHack On
    </Directory>
</VirtualHost>
```

Figure 20
XBitHack directive configuration

Save the changes using `:w` or save and exit the text-editor using the `:wq` command.

- 6. `sudo a2ensite ssi1.conf`
calls apache2 to enable the configuration of ssi1
- 7. `sudo service apache2 reload`
reloads the apache web server to cope with the configuration changes

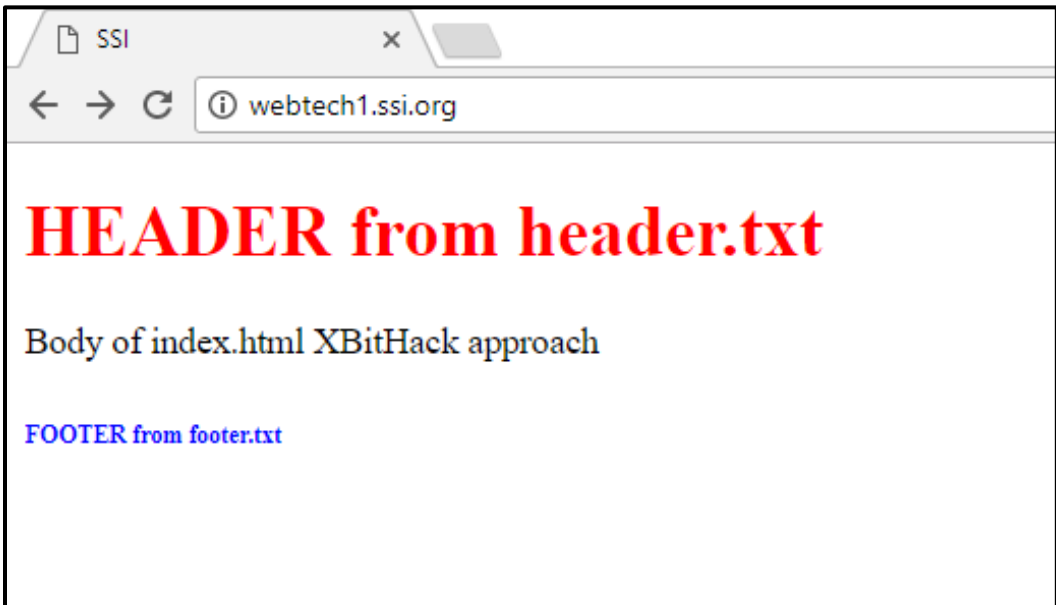


Figure 21
SSI Output - XBitHack

Filename extension-based

This method requires that the filename extension to be changed to *.shtml*

1. Please refer to the step one (1) above. But make sure to create an *index.shtml* file and not a *.html* one.
2. Please refer to step three (3) above.
3. Please refer to step four (4) above.
4. **sudo vi /etc/apache2/sites-available/ssi2.conf**
creates a blank configuration file using the vi text-editor

Input the commands shown in the figure below for the configuration.

Note: Lines beginning with a hash sign (#) are considered as comments

```
# VirtualHost configuration file for enabling
# SSI using the Filename Extension-based approach

<VirtualHost *:80>
    ServerName webtech1.ssi.org
    DocumentRoot /var/www/ssi2

    <Directory /var/www/ssi2>
        Options +Includes
        AllowOverride All
        AddType text/html .shtml
        AddOutputFilter INCLUDES .shtml
    </Directory>

    DirectoryIndex index.shtml
</VirtualHost>
```

Figure 22
Filename Extension-based approach

Save the changes using **:w** or save and exit the text-editor using the **:wq** command.

AddType – parses any file with a particular extension like *.shtml*

AddOutputFilter – parses any file with a particular extension like *.shtml*

DirectoryIndex – specifies what files to look for when a site request is made

5. **sudo a2ensite ssi2.conf**
calls apache2 to enable the configuration of ssi2
6. Please refer to step seven (7) provided above

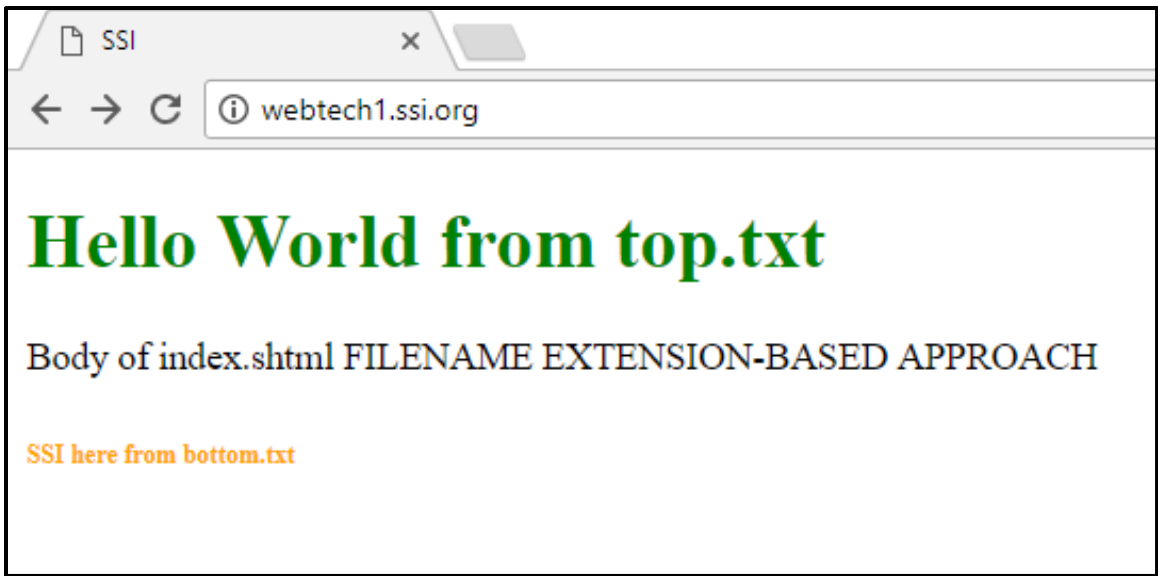


Figure 23
SSI Output – Filename Extension-based

HTTPS REDIRECT WITH SELF-SIGNED CERTIFICATE

When hosting a website, it is important that your website is secured. We can achieve this by using the OpenSSL certificate to encrypt our site to HTTPS.

To do this, we provide you with a quick guide:

1. **a2enmod ssl**
this command calls the apache2 and tells it to enable it's SSL module
2. **sudo service apache2 restart**
to enable the new configuration, we need to restart apache
3. **sudo openssl genrsa -out /etc/ssl/private/secure.key 2048**
this command generates the RSA that would be inside the secure.key.
2048 is the number of bits that would be generated for the encryption of
the key, the higher the bits the more difficult it becomes to decrypt the key
4. **sudo openssl req -new -key /etc/ssl/private/secure.key -out /etc/ssl/certs/secure.csr**
using the key that we have generated, we will now create our .csr file

This will now prompt you with some needed information.

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:PH
State or Province Name (full name) [Some-State]:Benguet
Locality Name (eg, city) []:Baguio city
Organization Name (eg, company) [Internet Widgits Pty Ltd]:SLU-SCIS
Organizational Unit Name (eg, section) []:WebTech Group3
Common Name (e.g. server FQDN or YOUR name) []:webtech1.secure.org
Email Address []:2161411@slu.edu.ph

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Figure 24
Self-Signed OpenSSL Certificate

5. `sudo openssl x509 -req -days 365 -in /etc/ssl/certs/secure.csr -
signkey /etc/ssl/private/secure.key -out
/etc/ssl/certs/secure.crt`

this command will create the certificate file

x509 – specifies that we want to create a self-signed certification and not
generating a certificate signing request

-days 365 – this specifies the number of days that the certificate is valid,
for this project it is set to 365 days or 1 year

6. `sudo vi /etc/apache2/sites-available/webtech1.secure.conf`
this opens the vi text-editor to input the configuration that will be used

Input the commands shown in the picture below for the configuration.

Note: Lines beginning with a hash sign (#) are considered as comments.

```
# VirtualHost configuration file for redirecting the website request
# to an HTTPS one with the use of OpenSSL

<VirtualHost *:80>
    ServerName webtech1.secure.org
    Redirect permanent / https://webtech1.secure.org
</VirtualHost>
<VirtualHost *:443>
    ServerName webtech1.secure.org
    DocumentRoot /var/www/security
    SSLEngine on
    SSLCertificateFile      /etc/ssl/certs/secure.crt
    SSLCertificateKeyFile   /etc/ssl/private/secure.key
</VirtualHost>
```

Figure 25
Configuration commands for the virtual host

Save the changes using `:w` or save and exit the text-editor using the `:wq` command.

80 – the port number assigned to HTTP

443 – the port number assigned to Secure HTTP

Redirect permanent – permanently redirects the address as an HTTPS request

SSLEngine on – enables or disables SSL for the virtual host

SSLCertificateFile – points out where the SSL certificate file is located

SSLCertificateKeyFile – points out where the SSL key file is located

7. **sudo a2ensite webtech1.secure.conf**
enables the configuration file for *webtech1.secure.org*

8. **sudo service apache2 reload**
reloads the apache web server to cope with the changes made in the server.

Note: Self-signed SSL certificates return unsecure websites because they have not been validated by a trusted Certificate Authority.

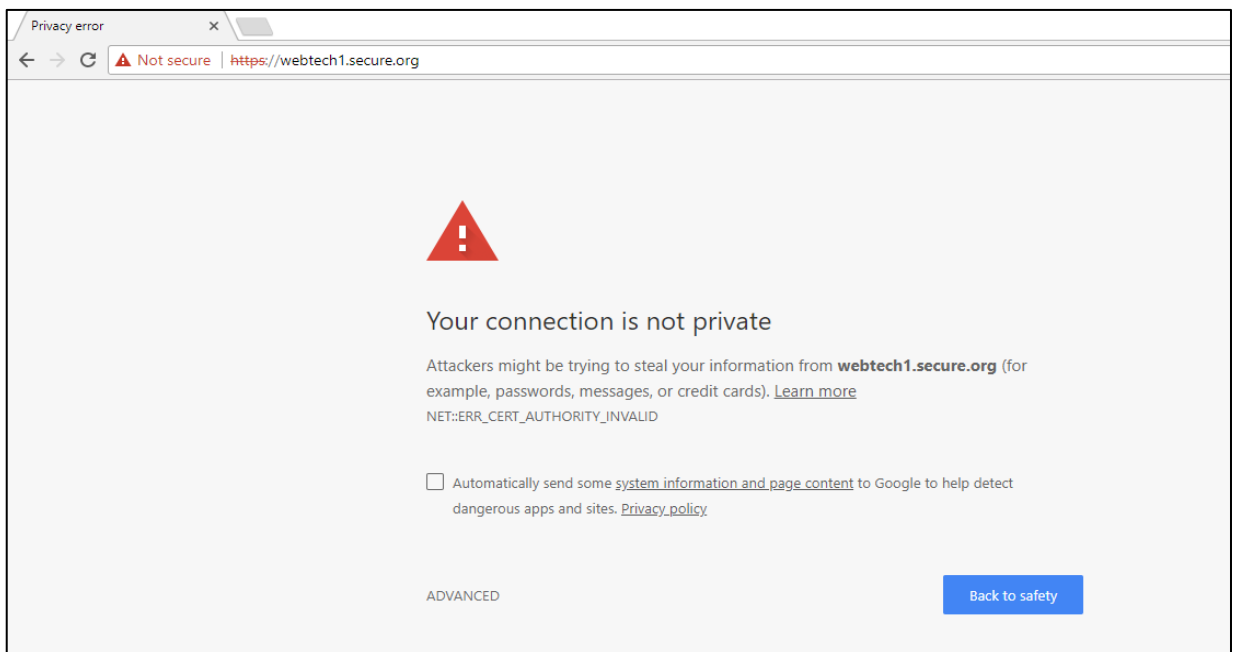


Figure 26
Test-Encryption – Google Chrome

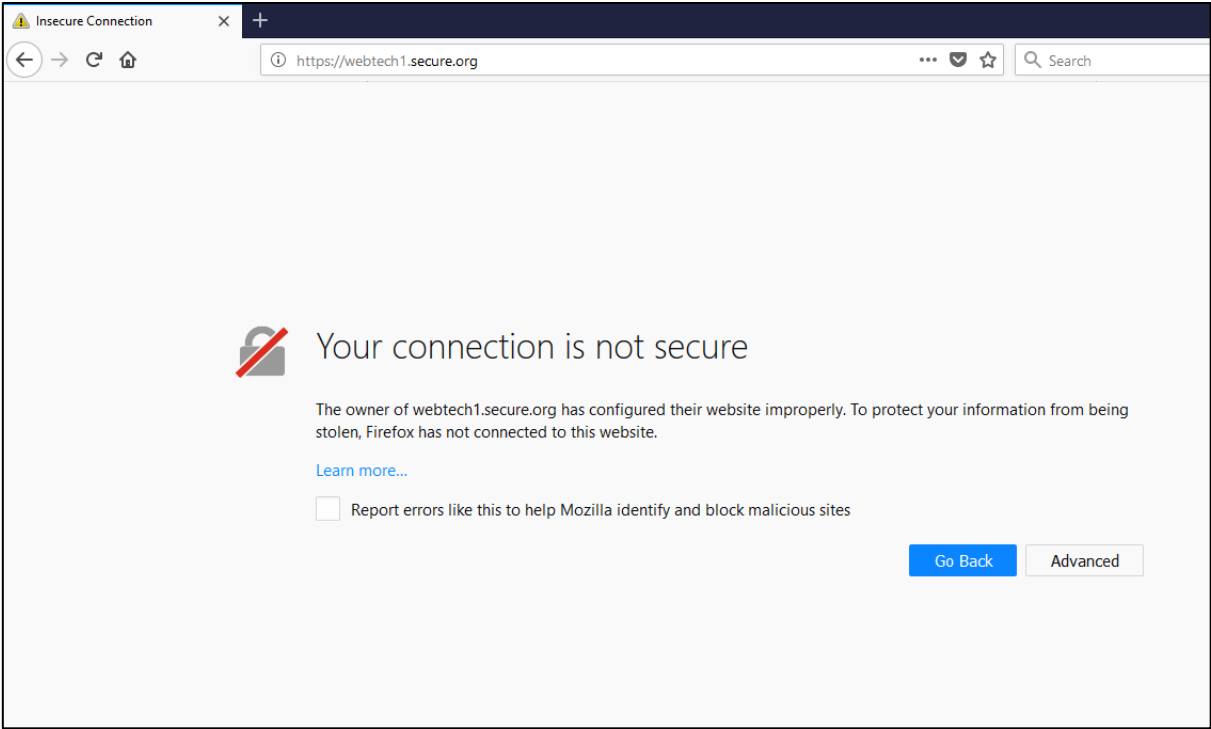


Figure 27
Test-Encryption – Mozilla Firefox