## Debian Based:

**Quick References:**

**Default ServerConfig**: /etc/apache2/apache2.conf

**Default ServerRoot:** /etc/apache2

**Default DocumentRoot:** /var/www/html/

**Quick Commands:**

$ apachectl {restart|start|stop}

$ service apache2 {restart|stop|start}

$ systemctl {restart|stop|start} apache2.service

$ a2ensite, a2dissite, a2enmod, a2dismod

## RedHat /Arch Based:

**Default ServerConfig**: /etc/httpd/conf.d/httpd.conf

**Default ServerRoot:** /etc/httpd

**Default DocumentRoot:** /var/www/html/

**Quick Commands:**

$ apachectl {restart|start|stop}

$ service httpd{restart|stop|start}

$ systemctl {restart|stop|start} httpd.service

**Find DocumentRoot**

Setup Mod\_Sec. Not matter what happens we will have to install mod\_sec with owasp rules. If it breaks the site. Just go error by error and bypass it in rule engine. If we get a bunch of nginx servers. All we have to do is install apache , set up rules and set up that proxy. So lets set up mod\_sec on centos/Redhat

## Mysql :

Create a dedicated user for the database. We are going say its wordpress for this checklist

$ mysql -u root -p

CREATE DATABASE wordpress;

CREATE USER wordpressuser@localhost IDENTIFIED BY 'password';

GRANT ALL PRIVILEGES ON wordpress.\* TO wordpressuser@localhost IDENTIFIED BY 'password';

FLUSH PRIVILEGES;

## Gentoo:

**Default Server Config**: /etc/apache2/httpd.conf

**Default Server Root:** /etc/apache2

**Default Document Root:** /var/www/localhost/htdocs/

# Nginx

Reverse Proxy with Apache TO nginx.

CentOS/Redhat

In case we get an nginx server, instead up setting up mod\_security for nginx, We are gonna place httpd in front of nginx and configure mod\_sec for nginx. We will place apache on port 8080 and proxy it into port 80 and 443, since those are the ports nginx will be scored on. We will also drop all inbound connections to port 80 and 443. This is okay because apache will proxy to it. That way they can't even hit port 80 and 443. THIS IS MY UNDERSTANDING. Fix anything that i goofed up on. We are gonna start off with redhat and move on to debian.

We need a few modules to work . these should be enabled by default on httpd fresh install.

**mod\_proxy,** the main proxy module Apache module for redirecting connections; it allows Apache to act as a gateway to the underlying application servers.

**mod\_proxy\_http**, which adds support for proxying HTTP connections.

**mod\_proxy\_balancer** and **mod\_lbmethod\_byrequests**, which add load balancing features for multiple backend servers.

Check them by. You should see them enabled

$ sudo httpd -M |grep proxy && sudo httpd -M |grep lbmethod

proxy\_module (shared)

lbmethod\_byrequests\_module (shared)

proxy\_balancer\_module (shared)

proxy\_http\_module (shared)

You should be good to go.. But if you are not edit the proxy conf and uncomment the following lines

$ sudo nano /etc/httpd/conf.modules.d/00-proxy.conf

/etc/httpd/conf.modules.d/00-proxy.conf

LoadModule proxy\_module modules/mod\_proxy.so

LoadModule lbmethod\_byrequests\_module modules/mod\_lbmethod\_byrequests.so

LoadModule proxy\_balancer\_module modules/mod\_proxy\_balancer.so

LoadModule proxy\_http\_module modules/mod\_proxy\_http.so

Now let's setup httpd to redirect. We are assuming nginx is already loaded. Is this is why we are doing this. This will also work with any other server. Such as tomcat. Just use common sense and translate it to a tomcat installation.

**First let's change nginx port to 8080. Find the server block that is using nginx and change the port to 8080. This example uses the default.**

**sudo emacs /etc/nginx/nginx.conf**

include /etc/nginx/conf.d/\*.conf;

server {

listen 8080 default\_server;

listen [::]:8080 default\_server;

#server\_name \_;

server\_name 138.68.232.77;

#root /usr/share/nginx/html;

root /usr/share/nginx/html/drupal;

# Load configuration files for the default server block.

include /etc/nginx/default.d/\*.conf;

Now if we start apache it should be listening on port 80. Lets do that now before we edit the apache vhost and start redirecting traffic

$ service httpd start

**Now lets change the apache vhost to proxy and send the traffic to port 8080**

# OR which ever Vhost file is the default. If there is not default site. Add this block to the very bottom of the main **/etc/httpd/conf/httpd.conf.** Same results

**sudo nano /etc/httpd/conf.d/default-site.conf**

<VirtualHost \*:80>

ProxyPreserveHost On

ProxyPass / http://127.0.0.1:8080/

ProxyPassReverse / http://127.0.0.1:8080/

</VirtualHost>

Listen 443

NameVirtualHost \*:443

<VirtualHost \*:443>

SSLEngine On

# Set the path to SSL certificate

# Usage: SSLCertificateFile /path/to/cert.pem

SSLCertificateFile /etc/apache2/ssl/file.pem

# Servers to proxy the connection, or;

# List of application servers:

# Usage:

# ProxyPass / http://[IP Addr.]:[port]/

# ProxyPassReverse / http://[IP Addr.]:[port]/

# Example:

ProxyPass / http://0.0.0.0:8080/

ProxyPassReverse / http://0.0.0.0:8080/

# Or, balance the load:

# ProxyPass / balancer://balancer\_cluster\_name

</VirtualHost>

Restart apache and you should now have successfully redirected traffic from port 80 to 8080

From apache 80 to nginx 8080

**$ service httpd restart**

**$ service nginx restart**