**Firewalls**

For this lab assignment, I would be writing a firewall policy which allows access to HTTP, HTTPS and SSH. Additionally, I will be redirecting the incoming HTTP and HTTPS requests in the server container. I am also required to redirect incoming requests to port 2222 to the client container’s SSH service.

**Prerequisites:**

First, I installed Nginx with **libModSecurity** and OWASP core rule set inside the server container. I used the command below to install all the required dependencies:

**apt-get install autoconf automake build-essential bzip2 checkinstall devscripts flex g++ gcc git graphicsmagick-imagemagick-compat graphicsmagick-libmagick-dev-compat libaio-dev libaio1 libass-dev libatomic-ops-dev libavcodec-dev libavdevice-dev libavfilter-dev libavformat-dev libavutil-dev libbz2-dev libcurl4-openssl-dev libfaac-dev libfreetype6-dev libgd-dev libgeoip-dev libgeoip1 libgif-dev libgpac-dev libgsm1-dev libjack-jackd2-dev libjpeg-dev libjpeg-progs libjpeg8-dev liblmdb-dev libmp3lame-dev libncurses5-dev libopencore-amrnb-dev libopencore-amrwb-dev libpam0g-dev libpcre3 libpcre3-dev libperl-dev libpng12-dev libpng12-0 libpng12-dev libreadline-dev librtmp-dev libsdl1.2-dev libssl-dev libssl1.0.0 libswscale-dev libtheora-dev libtiff5-dev libtool libva-dev libvdpau-dev libvorbis-dev libxml2-dev libxslt-dev libxslt1-dev libxslt1.1 libxvidcore-dev libxvidcore4 libyajl-dev make openssl perl pkg-config tar texi2html unzip zip zlib1g-dev**

Then, I used the following command to delete the existing Nginx:

* **apt purge nginx**

The next step was to clone the ModSecurity source from its official Git repository and build the libModSecurity. To achieve this, the following commands were then executed in the given order:

* **cd /opt/**
* **git clone https://github.com/SpiderLabs/ModSecurity**
* **cd ModSecurity**
* **git checkout -b v3/master origin/v3/master**
* **sh build.sh**
* **git submodule init**
* **git submodule update**
* **./configure**

In order to compile the above configuration, I used the command:

* **make && make install.**

I downloaded the ***ModSecurity Nginx*** connector by switching back to the “opt” directory as follows:

* **git clone https://github.com/SpiderLabs/ModSecurity-nginx.git**

To download the latest version of Nginx, in the “opt” directory, use:

* **wget**[**http://nginx.org/download/nginx-1.12.2.tar.gz**](http://nginx.org/download/nginx-1.12.2.tar.gz)**.** Then unzip this downloaded file using tar.

Compile and install nginx using:

* **cd nginx-1.12.2**
* **./configure --user=www-data --group=www-data --with-pcre-jit --with-debug --with-http\_ssl\_module --with-http\_realip\_module --add-module=/opt/ModSecurity-nginx**
* **make && make install**

The ModSecurity source code that I downloaded earlier includes a sample ModSecurity.conf file with some recommended settings, so I copied this file to the folder with the nginx configuration files:

* **cp /opt/ModSecurity/modsecurity.conf-recommended/usr/local/nginx/conf/modsecurity.conf**

Then I created a symlink from the nginx binary to my executable path using:

* **ln -s /usr/local/nginx/sbin/nginx /bin/nginx**

Next, I opened up the Nginx global configuration file to make the following changes in the config file:

* **nano /usr/local/nginx/conf/nginx.conf**

**I added the following lines of code below in the Nginx global configuration file:**

**user www-data;**

**pid /var/run/nginx.pid;**

**events {**

**worker\_connections 1024;**

**}**

**http {**

**server {**

**listen 80 default\_server;**

**listen [::]:80 default\_server;**

**server\_name \_;**

**return 301 https://$host$request\_uri;**

**}**

**server {**

**listen 443 ssl default\_server;**

**listen [::]:443 ssl default\_server;**

**ssl\_certificate /etc/ssl/certs/server.pokhriyal.ccd.lab.pem;**

**ssl\_certificate\_key /etc/ssl/private/server.pokhriyal.ccd.lab-key.pem;**

**root html;**

**index index.php index.html index.htm index.nginx-debian.html;**

**server\_name server.pokhriyal.ccd.lab;**

**location = /favicon.ico { log\_not\_found off; access\_log off; }**

**location = /robots.txt { log\_not\_found off; access\_log off; allow all; }**

**location ~\* \.(css|gif|ico|jpeg|jpg|js|png)$ {**

**expires max;**

**log\_not\_found off;**

**}**

**location / {**

**try\_files $uri $uri/ /index.php?$args;**

**}**

**location ~ \.php$ {**

**#include snippets/fastcgi-php.conf;**

**fastcgi\_pass unix:/var/run/php/php7.2-fpm.sock;**

**fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;**

**}**

**location ~ /\.ht {**

**deny all;**

**}**

**}**

**}**

I created a nginx.service file to create a systemd service for nginx:

**nano /etc/systemd/system/nginx.service**

In the above file: **nginx.service**, I added the following lines of code:

**[Unit]**

**Description=The NGINX HTTP and reverse proxy server**

**After=syslog.target network.target remote-fs.target nss-lookup.target**

**[Service]**

**Type=forking**

**PIDFile=/var/run/nginx.pid**

**ExecStartPre=/bin/nginx -t**

**ExecStart=/bin/nginx**

**ExecReload=/bin/kill -s HUP $MAINPID**

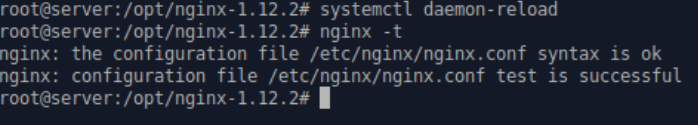
**ExecStop=/bin/kill -s QUIT $MAINPID**

**PrivateTmp=true**

**[Install]**

**WantedBy=multi-user.target**

I then executed **systemctl daemon-reload** to take effect. If the Nginx configuration is proper, you can see the following message:

Next, I installed OWASP Core Rule Set using the below commands:

* **cd /opt/**
* **git clone https://github.com/SpiderLabs/owasp-modsecurity-crs.git**
* **cd owasp-modsecurity-crs/**
* **cp -R rules/ /usr/local/nginx/conf/**
* **cp /opt/owasp-modsecurity-crs/crs-setup.conf.example /usr/local/nginx/conf/crs-setup.conf**

I edited **modsecurity.conf** file to include the following OWASP rule set files:

**#Load OWASP Config**

**Include crs-setup.conf**

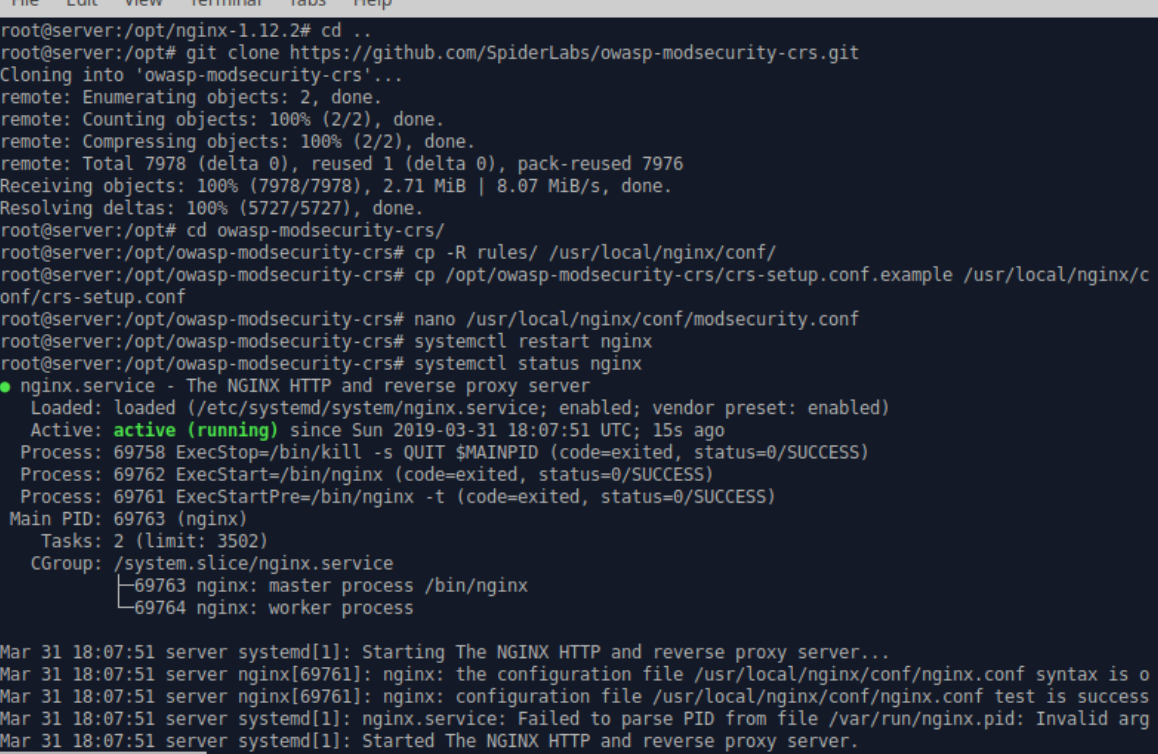
**#Load all other Rules**

**Include rules/\*.conf**

**#Disable rule by ID from error message**

**#SecRuleRemoveById 920350**

Finally, I restarted Nginx to take effect:



Now, I begin the process of setting up a Firewall using Iptables on my host ubuntu. Inside the Virtual Machine, I used iptables-persistent package to save the iptables rules in order to survive a reboot as follows:

* **sudo apt-get install iptables-persistent**

I want my VM to accept all traffic on my loopback interface. To do this, I ran the following commands:

* **sudo iptables -A INPUT -i lo -j ACCEPT**
* **sudo iptables -A OUTPUT -o lo -j ACCEPT**

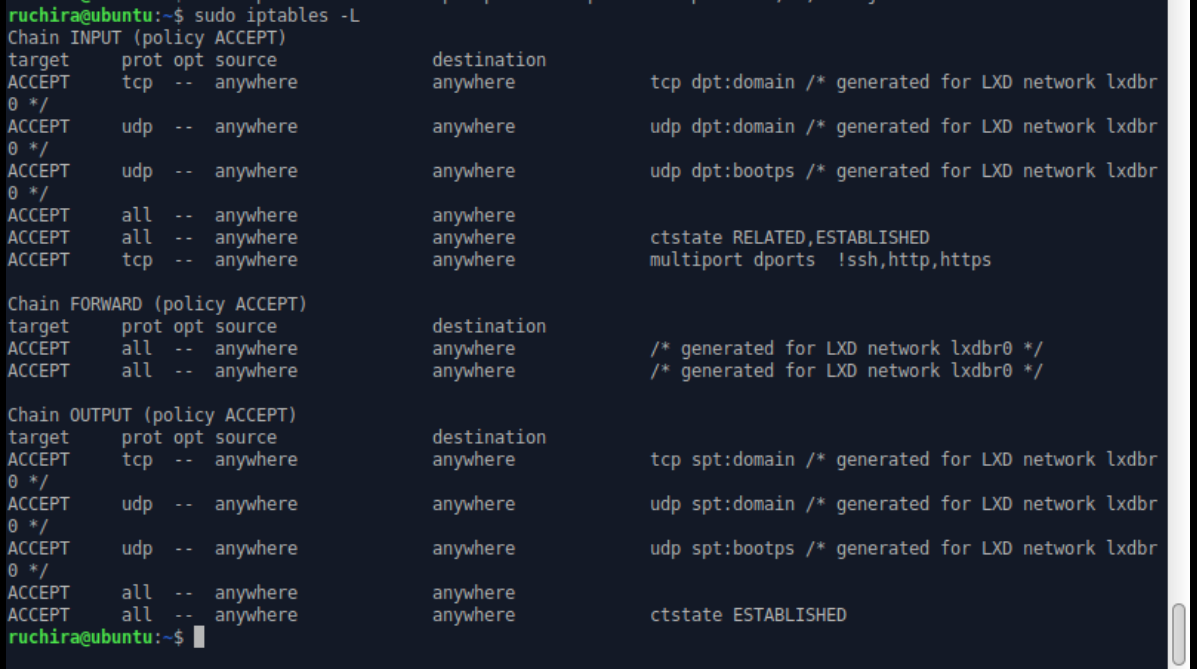
Next, we need to create a firewall rule that allows established and related incoming/outgoing traffic. This can be done using the following two commands:

* **sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT**
* **sudo iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT**

To allow access to SSH, HTTP and HTTPS, I used the following command:

* **iptables -A INPUT -p tcp -m multiport ! --dports 22,80,443 -j ACCEPT**

It’s evident from the following screenshot that a firewall policy is developed which allows access to SSH, HTTP and HTTPS:

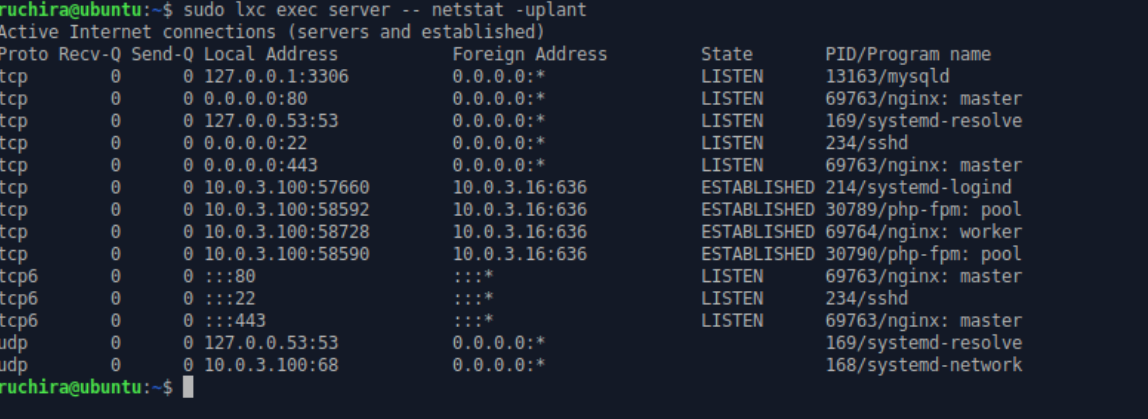


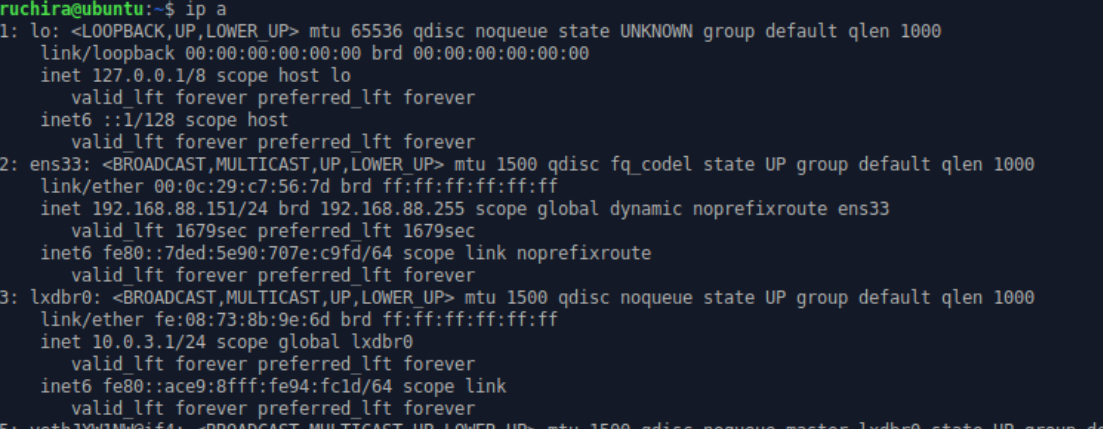
**Firewall Rules for Prerouting:**

* **iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to 10.0.3.100:80**
* **iptables -t nat -A PREROUTING -p tcp --dport 443 -j DNAT --to 10.0.3.100:443**
* **iptables -t nat -A PREROUTING -p tcp --dport 2222 -j DNAT --to 10.0.3.101:22**

Using the above, we are forwarding incoming HTTP and HTTPS requests to the server container whose IP address is 10.0.3.100. These rules specify that the NAT table use the built-in PREROUTING chain to forward incoming HTTP and HTTPS requests exclusively to the server container.

All HTTP connections to port 80 and HTTPS connections to port 443 are routed to the server container.



  
Using the IP of the VM, I logged into the website from a browser outside my VM.

This IP address then redirected to the VM container and again redirected the website to https, which is what I setup in the first server block of Nginx, proving that the configuration is up and working correctly. Following is a screenshot showing this:

