Installing NGINX with ModSecuirty and reverse proxy to Express server

Setting up Express with nginx and pm2

Installation of Node

Express is built on Node, so we must first install Node.

```
$ sudo apt-get update
$ curl -sL https://deb.nodesource.com/setup_0.12 | sudo bash -
$ sudo apt-get install -y nodejs
```

Express

Now we need to set up Express. Express provides us with a Generator. First, we install the Express generator tool.

```
$ npm install express-generator -g
```

This allows us to use the command `express [name]`

```
$ express firingrange
```

This will give us the following structure

```
create : firingrange
create : firingrange/package.json
create : firingrange/app.js
create : firingrange/public
create : firingrange/public/javascripts
create : firingrange/public/images
create : firingrange/public/stylesheets
create : firingrange/public/stylesheets/style.css
create : firingrange/routes
create : firingrange/routes/index.js
create : firingrange/routes/users.js
create : firingrange/views
create : firingrange/views/index.jade
create : firingrange/views/layout.jade
create : firingrange/views/error.jade
create : firingrange/bin
create : firingrange/bin/www
```

```
install dependencies:
    $ cd firingrange && npm install

run the app:
    $ DEBUG=firingrange:* ./bin/www
```

Installation of pm2

```
$ npm install -g pm2
```

Instead of running the command *node*, we simply run the command *pm2 start*. So instead of

```
$ node bin/appfiringrange
```

We will run

```
$ pm2 start bin/appfiringrange
```

This will be print

```
┌──────────────┬────┬──────┬──────┬────────┬─────────┬────────┬────────────┬──────────┐
│ App name     │ id │ mode │ pid  │ status │ restart │ uptime │ memory     │
watching │
├──────────────┼────┼──────┼──────┼────────┼─────────┼────────┼────────────┤
│ firingrange │ 0  │ fork │ 2854 │ online │ 6621    │ 0s     │ 7.102 MB   │
disabled │
└──────────────┴────┴──────┴──────┴────────┴─────────┴────────┴────────────┴──────────┘
```

If you go to `http://[YOUR IP]:3000/`, you'll see the same application running again. Only this time, it will stay running even if you log out, or the application had a temporary error, it will restart.

In the table above, you can see under the column header 'restart', the number of times the application has been restarted. You can run `pm2 list` later on to see if the application has been restarted since the last time you checked on it

Compiling Nginx with ModSecurity

We wanted to test ModSecurity with the OWASP ModSecurity ruleset on an Nginx reverse proxy in front of a node.js server.

Using ModSecurity with Nginx requires

1. compiling ModSecurity from source as a standalone module, and then
2. compiling Nginx from source with the ModSecurity module

Installation of dependencies

With the Nginx configuration options that we use below we will need to install the following dependencies on the system:

```
$ apt-get install build-essential libpcre3 libpcre3-dev libssl-dev
libtool autoconf apache2-prefork-dev libxml2-dev libcurl4-openssl-dev
```

Compiling ModSecurity

```
$ cd /usr/src
$ wget https://github.com/SpiderLabs/ModSecurity/archive/v2.8.0.tar.gz
$ tar -zxvf v2.8.0.tar.gz
```

Now compile ModSecurity as a standalone module that we can include when we build Nginx:

```
$ cd /usr/src/ModSecurity-2.8.0/
$ ./autogen.sh
$ ./configure --enable-standalone-module
$ make
```

Compiling Nginx

The mainline version is being used as per the recommendations of Nginx.

Download and extract the Nginx source code:

```
$ cd /usr/src
$ wget http://nginx.org/download/nginx-1.7.9.tar.gz
$ tar -zxvf nginx-1.7.9.tar.gz
```

Now we need to compile

```
$ cd /usr/src/nginx-1.7.9/
$ ./configure \
  --user=www-data \
  --group=www-data \
  --with-pcre-jit \
  --with-debug \
  --with-ipv6 \
  --with-http_ssl_module \
  --add-module=/usr/src/ModSecurity-2.8.0/nginx/modsecurity
```

The output from *configure* shows that we are using the nginx defaults:

```
nginx path prefix: "/usr/local/nginx"
nginx binary file: "/usr/local/nginx/sbin/nginx"
nginx configuration prefix: "/usr/local/nginx/conf"
nginx configuration file: "/usr/local/nginx/conf/nginx.conf"
nginx pid file: "/usr/local/nginx/logs/nginx.pid"
nginx error log file: "/usr/local/nginx/logs/error.log"
nginx http access log file: "/usr/local/nginx/logs/access.log"
nginx http client request body temporary files: "client_body_temp"
nginx http proxy temporary files: "proxy_temp"
nginx http fastcgi temporary files: "fastcgi_temp"
nginx http uwsgi temporary files: "uwsgi_temp"
nginx http scgi temporary files: "scgi_temp"
```

We can now compile and build Nginx:

```
$ make
$ make install
```

Init Script

Jason Giedymin's Nginx init script is used to manage the nginx service on system:

```
$ wget https://raw.github.com/JasonGiedymin/nginx-init-
ubuntu/master/nginx -O /etc/init.d/nginx
$ chmod +x /etc/init.d/nginx
$ update-rc.d nginx defaults
```

This script provides the following options for managing the Nginx service:

```
$ service nginx start|stop|restart|force-reload|reload|
  status|configtest|quietupgrade|terminate|destroy
```

Nginx Configuration

There are a number of configuration options in nginx.conf that you may want to modify on a production system.

We like to keep virtual hosts in their own directories, similar to the way it works when you install Nginx using apt-get. We create two directories to hold the virtual host files:

```
$ mkdir /usr/local/nginx/conf/sites-available
$ mkdir /usr/local/nginx/conf/sites-enabled
```

Remove the sample virtual hosts from nginx.conf (make a backup copy first), and then add the following line to the end of nginx.conf, before the closing curly brace:

```
include /usr/local/nginx/conf/sites-enabled/*;
```

We can now create the virtual hosts in the sites-available directory and then link them in the sites-enabled directory as required. We are using a simple virtual host similar to the example in nginx.conf for testing purposes:

```
server {
  listen       80;
  server_name  example.com;

  location / {
    root   html;
    index  index.html index.htm;
  }
}
```

Configure ModSecurity & reverse proxy

Download and extract the current version of the OWASP ruleset:

```
$ cd /usr/src
$ wget https://github.com/SpiderLabs/owasp-modsecurity-
crs/tarball/master -O owasp.tar.gz
$ tar -zxvf owasp.tar.gz
```

The OWASP rules are extracted to the **/usr/src/SpiderLabs-owasp-modsecurity-crs-ebe8790/** folder. (The last part of the folder name depends on the ruleset version and will most likely be different on your system).

Edit your virtual hosts file to enable ModSecurity:

```
server {
  listen       80;

  location / {
    ModSecurityEnabled on;
    ModSecurityConfig modsecurity.conf;
    proxy_pass http://162.209.99.114:8080/;
         // The proxy ip is the address where we want to reverse proxy
      which is
       //at 8080 port
}
}
```

The ModSecurity source code that we downloaded earlier includes a sample modsecurity.conf file with some recommended settings. Copy this file to the folder with the Nginx configuration files (**/usr/local/nginx/conf** in our case):

```
$ cp /usr/src/ModSecurity-2.8.0/modsecurity.conf-recommended \
  /usr/local/nginx/conf/modsecurity.conf
```

We also need a unicode mapping file that is being referenced in the recommended modsecurity.conf file:

```
$ cp /usr/src/ModSecurity-2.8.0/unicode.mapping /usr/local/nginx/conf/
```

The OWASP rules we downloaded above contains three types of files that we need:

- A base configuration example file, modsecurity*crs*10_setup.conf.example
- Several .conf rule files
- Several supporting .data files

With Apache you would normally include the .conf files into a master configuration file, but ModSecurity for Nginx does not allow includes so the only option is to append the individual .conf files to our modsecurity.conf file. We should then copy all the .data files to the directory where modsecurity.conf is located:

```
$ cd /usr/src/SpiderLabs-owasp-modsecurity-crs-ebe8790/
$ cat modsecurity_crs_10_setup.conf.example >> \
  /usr/local/nginx/conf/modsecurity.conf
$ cd /usr/src/SpiderLabs-owasp-modsecurity-crs-ebe8790/base_rules
$ cat *.conf >> /usr/local/nginx/conf/modsecurity.conf
$ cp *.data /usr/local/nginx/conf/
```

Restart Nginx to enable ModSecurity on the virtual host