**Lecture 29**

Nginx-Source Code Installation-Cloud-Advancement-Track Demo (Live Session 26November 2022)

Nginx solution reverse proxy

**Nginx server installation**

========================================

Compiling and Installing from Source - This is without WAF

========================================

Compiling NGINX Open Source from source affords more flexibility than pre-built packages: you can add particular modules (from NGINX or third parties), and apply latest security patches

Download & Installing NGINX Dependencies

========================================

Prior to compiling NGINX Open Source from source, you need to install libraries for its dependencies:

Option 01: Download offline packages (gunzip files - i have uploaded the files on portal)

-------------------------------------------------------------

1. pcre-8.44.tar.gz

2. zlib-1.2.11.tar.gz

3. openssl-3.0.2.tar.gz

4. nginx-1.20.2.tar.gz

Dependency 00: Install pre-defined packages

========================================

# yum install wget gcc-c++ perl-core curl

Dependency 01: PCRE – Perl Compatible Regular Expressions. Required by the NGINX Core and Rewrite modules.

========================================

# mkdir /apps/pcre

# cd /apps/pcre

# tar -zxf /apps/packages/pcre-8.44.tar.gz

# cd pcre-8.44

# ./configure

# make

# make install

Dependency 02: zlib – Supports header compression. Required by the NGINX Gzip module.

========================================

# mkdir /apps/zlib

# cd /apps/zlib

# tar -zxf /apps/packages/zlib-1.2.11.tar.gz

# cd zlib-1.2.11

# ./configure

# make

# make install

Dependency 03: OpenSSL – Supports the HTTPS protocol. Required by the NGINX SSL module and others.

========================================

# mkdir /apps/openssl

# cd /apps/openssl

# tar -zxf /apps/packages/oopenssl-3.0.2.tar.gz

# cd openssl-3.0.2

# ./Configure linux-x86\_64 --prefix=/usr

# make

# make install

Install nginx using source without modesecurity WAF

========================================

# mkdir /apps/nginx

# cd /apps/nginx

# tar -zxf /apps/packages/nginx-1.20.2.tar.gz

# cd nginx-1.20.2

# ./configure --prefix=/apps/nginx --with-pcre=/apps/pcre-8.44 --with-zlib=/apps/zlib-1.2.11 --with-http\_ssl\_module --with-mail --with-mail\_ssl\_module --with-http\_stub\_status\_module --without-http\_auth\_basic\_module --without-http\_browser\_module --without-http\_charset\_module --without-http\_empty\_gif\_module --without-http\_fastcgi\_module --without-http\_geo\_module --without-http\_map\_module --without-http\_memcached\_module --without-http\_referer\_module --without-http\_scgi\_module --without-http\_ssi\_module --without-http\_split\_clients\_module --without-http\_userid\_module --without-http\_uwsgi\_module

# make

# make install

-----------------------------------------------------------------------------------------------------------------

Installing Let’s Encrypt SSL on Linux with Certbot to use secure connection --- This is optional if you want to use port 443 -

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/SSL-on-amazon-linux-2.html -- This will only work if you have registered domain

-----------------------------------------------------------------------------------------------------------------

a. Download the Extra Packages for Enterprise Linux (EPEL) 7 repository packages. These are required to supply dependencies needed by Certbot.

sudo wget -r --no-parent -A 'epel-release-\*.rpm' https://dl.fedoraproject.org/pub/epel/7/x86\_64/Packages/e/

b. Install the repository packages as shown in the following command.

sudo rpm -Uvh dl.fedoraproject.org/pub/epel/7/x86\_64/Packages/e/epel-release-\*.rpm

c. Enable EPEL as shown in the following command.

sudo yum-config-manager --enable epel\*

d. You can confirm that EPEL is enabled with the following command.

sudo yum repolist all

e. Install and run Certbot - Install the Amazon Extras repo for epel.

sudo amazon-linux-extras install epel -y

f. Install Certbot packages and dependencies using the following command.

sudo yum install -y certbot python2-certbot-nginx

g. Create SSL certs for all domains

sudo certbot --nginx

h. Certbot displays the Common Name and Subject Alternative Name (SAN) that we provided in your nginx conf file server\_name apps.hackercloud.in ( DONT USE HACKERCLOUD.IN - USE YOUR OWN DOMAIN NAME)

hit Enter

I.select install certificate - Certbot completes the configuration of Nginx and reports success and other information.

After you complete the installation, test your site.

----------------------------------------------------------------------------------------------

Configure automated certificate renewal

refer cron steps

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/SSL-on-amazon-linux-2.html

----------------------------------------------------------------------------------------------

======================================================

Creation of nginx user without root privileges and no home and login option

======================================================

# adduser --system --no-create-home --user-group -s /sbin/nologin nginx

======================================================

NGINX Hardening

Change Server Header Banner

===========================

# Default Nginx configuration will expose server information with its version, which is highly recommended to mask it if you are working in a PCI-DSS environment

======================================================

Edit nginx.config and add nginx user

======================================================

# vi /apps/nginx/conf/nginx.conf

user nginx;

======================================================

Disable nginx server\_tokens

======================================================

By default, the server\_tokens directive in nginx displays the nginx version number. It is directly visible in all

automatically generated error pages but also

present in all HTTP responses in the Server header.

This could lead to information disclosure – an unauthorized user could gain knowledge about the version of nginx that you

use. You should disable the

server\_tokens directive in the nginx configuration file by setting

server\_tokens off;

======================================================

Control Resources and Limits

======================================================

To prevent potential DoS attacks on nginx, you can set buffer size limitations for all clients. You can do this in the nginx

configuration file using the

following directives:

client\_body\_buffer\_size

-----------------------

use this directive to specify the client request body buffer size. The default value is 8k or 16k but it is recommended to set this as low as 1k:

client\_body\_buffer\_size 1k.

client\_max\_body\_size

--------------------

use this directive to specify the maximum accepted body size for a client request. A 1k directive should be sufficient but you need to increase it if you are receiving file uploads via the POST method.

large\_client\_header\_buffers

---------------------------

use this directive to specify the maximum number and size of buffers to be used to read large client request headers. A large\_client\_header\_buffers 2 1k directive sets the maximum number of buffers to 2, each with a maximum size of 1k. This directive will accept 2 kB data URI.

Note: Some sources suggest that setting such limits may prevent potential buffer overflow attacks if such vulnerabilities are found in nginx.

----------------------------------------------------------------------------------------------

Set Up and Configure nginx Access and Error Logs

----------------------------------------------------------------------------------------------

The nginx access and error logs are enabled by default and are located in logs/error.log and logs/access.log respectively. If

you want to change the location, you can use the error\_log directive in the nginx configuration file. You can also use this directive to specify the logs that

will be recorded according to their severity level. For example, a crit severity level will cause nginx to log critical issues and all issues that have a higher severity level than crit. To set the severity level to crit, set the error\_log directive as follows:

error\_log logs/error.log crit;

You can find a complete list of error\_log severity levels in official nginx documentation.

You can also modify the access\_log directive in the nginx configuration file to specify a non-default location for access

logs. Finally, you can use the

log\_format directive to configure the format of the logged messages as explained in nginx documentation.

----------------------------------------------------------------------------------------------

Configure Nginx to Include Security Headers

----------------------------------------------------------------------------------------------

To additionally harden your nginx web server, you can add several different HTTP headers. Here are some of the options that we recommend.

X-Frame-Options

---------------

You use the X-Frame-Options HTTP response header to indicate if a browser should be allowed to render a page in a <frame> or an <iframe>. This could prevent clickjacking attacks. Therefore, we recommend that you enable this option for your nginx server.

To do this, add the following parameter to the nginx configuration file in the server section:

add\_header X-Frame-Options "SAMEORIGIN" always;

Strict-Transport-Security

--------------------------

HTTP Strict Transport Security (HSTS) is a method used by websites to declare that they should only be accessed using a secure connection (HTTPS). If a website declares an HSTS policy, the browser must refuse all HTTP connections and prevent users from accepting insecure SSL certificates. To add an HSTS header to your nginx server, you can add the following directive to your server section:

add\_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;

CSP and X-XSS-Protection

------------------------

Content Security Policy (CSP) protects your web server against certain types of attacks, including Cross-site Scripting attacks (XSS) and data injection attacks. You can implement CSP by adding the following example Content-Security-Policy header (note that the actual header should be configured to match your unique requirements):

add\_header Content-Security-Policy "default-src 'self' http: https: data: blob: 'unsafe-inline'" always;

The HTTP X-XSS-Protection header is supported by IE and Safari and is not necessary for modern browsers if you have a strong Content Security Policy. However, to help prevent XSS in the case of older browsers (that don’t support CSP yet), you can add the X-XSS Protection header to your server section:

add\_header X-XSS-Protection "1; mode=block" always;

---------------------------

Configure SSL and Cipher Suites

---------------------------

The default configuration of nginx allows you to use insecure old versions of the TLS protocol (according to the official documentation: ssl\_protocols TLSv1 TLSv1.1 TLSv1.2). This may lead to attacks such as the BEAST attack. Therefore, we recommend that you do not use old TLS protocols and change your configuration to support only newer, secure TLS versions.

To do this, add the following directive in the server section of the nginx configuration file:

# Enable TLSv1.2 and TLSv1.3, disable SSLv3.0, TLSv1.0 and TLSv1.1

ssl\_protocols TLSv1.2 TLSv1.3;

Additionally, you should specify cipher suites to make sure that no vulnerable suites are supported. To select the best cipher suites, read our article on TLS cipher hardening (https://www.acunetix.com/blog/articles/tls-ssl-cipher-hardening/) and add a ssl\_ciphers directive to the server section to select the ciphers (as suggested in the article on cipher hardening).

ssl\_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384;

----------------------------------------------------------------------------------------------

Secure Diffie-Hellman for TLS

----------------------------------------------------------------------------------------------

Diffie-Hellman is less secure than it was believed. One of the best practices lately added in a list is to secure Diffie-hellman. Generating unique DH GROUP and adding ssl\_dhparam in ssl.conf file does this.

Generate Unique DH Group by using OpenSSL

# cd ssl

# openssl dhparam -out dhparam.pem 4096

It will take few minutes and will generate a file dhparams.pem on a current working directory

Modify nginx.conf and add following in http block

ssl\_dhparam /apps/nginx/conf/ssl/dhparam.pem;

======================================================

CONFIGURE AS SERVICE

======================================================

Create a file as /lib/systemd/system/nginx.service

[Unit]

Description=The NGINX HTTP and reverse proxy server

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

Wants=network-online.target

[Service]

Type=forking

PIDFile=/apps/nginx/logs/nginx.pid

ExecStartPre=/apps/nginx/sbin/nginx -t

ExecStart=/apps/nginx/sbin/nginx

ExecReload=/apps/nginx/sbin/nginx -s reload

ExecStop=/bin/kill -s QUIT #MAINPID

PrivateTmp=true

[Install]

WantedBy=multi-user.target

Command to reload nginx service changes

---------------------------------------

systemctl daemon-reload

Nginx service commands

----------------------

systemctl start nginx

systemctl reload nginx

systemctl stop nginx

systemctl restart nginx

======================================================

Other Tools:

======================================================

SSL Scan

https://www.ssllabs.com/ssltest/index.html

----------------------------------------------------------------------------------------------

LOG ROTATION

----------------------------------------------------------------------------------------------

# vi /etc/logrotate.d/nginx

/apps/nginx/logs/\*log {

create 0644 nginx nginx

daily

rotate 10

missingok

notifempty

compress

sharedscripts

postrotate

/bin/kill -USR1 `cat /apps/nginx/logs/nginx.pid 2>/dev/null` 2>/dev/null || true

endscript

}

.