**Name of Title:** Learning Nginx

**Video Name:** 03\_02 Configuring Allow and Deny

**Estimated Length:**

**Author Name:** Michael Jenkins

**Chapter\_Section\_Video:**

**Video Objective:**

At the end of this video the learner will be able to use allow and deny directives to protect locations within the site.

**Introductory Statement:**

Only allow local IPs to access /admin; deny all others

**Script:**

SLIDE: 03\_02 Configuring Allow and Deny

Limiting access is one of the ways we can help secure the contents of a site. Fortunately, nginx provides the HTTP Access module that includes allow and deny directives. These are used to limit who is allowed to see certain content and who gets denied. Let’s take a look at how these directives are configured and how we can use them.

If you’re following along with the exercise files, you can use the Vagrantfile in the folder for this lesson. It will boot the VM and install nginx and all supporting software and data for our demo site.

If you're not using the exercise files, you can follow along with a VM running Ubuntu 18.04 LTS. You’ll need root access and you’ll need to install nginx.

Let’s start by considering a page in our demo site that we want to protect. One of the most sensitive pages on our site is the appointments page:

OPEN BROWSER TO <http://192.168.0.3/appointments/>

This page contains PII or Personally identifiable information about the vet clinics patients and their owners. This is information that we don’t want the general public to have access to.

So let’s configure nginx to first deny everyone from seeing this page.

The server is already up and running so I just need to connect to it.

Vagrant ssh

Sudo su -

vim /etc/nginx/conf.d/wisdompetmed.local.conf

As the root user, let’s edit the vhost for the site and create a location for the appointments directory:

location /appointments/ {

}

Now we can add the deny directive and use they keyword “all” to tell nginx not to serve this page to anyone:

location /appointments/ {

deny all;

}

As always, let’s test the configuration and then reload it.

SAVE FILE

Nginx -t

Systemctl reload nginx

Now let’s see what happens when we try to access the appointments location in our browser:

OPEN BROWSER

Should see 403 FORBIDDEN

With our deny all directive applied, nginx is refusing to serve the page and instead shows a page with 403 Forbidden.

This works, but we don’t want to have a web server that just forbids everyone from seeing the files it serves! Let's configure the location so that only requests coming from our internal network can see the appointments page.

vim /etc/nginx/conf.d/wisdompetmed.local.conf

Back in the vhost file, I’ll use the “allow” directive along with IP ranges to let nginx know what networks can access the site:

location /appointments/ {

# only allow IPs from the same network the server is on

allow 192.168.0.0/24;

allow 10.0.0.0/8;

deny all;

}

For the addresses, the notation we’re using here is called CIDR notation or Classless Inter-Domain Routing notation. Its a shorthand way to represent a range of IP addresses.

One thing I need to point out is that allow and deny directives will be applied in the order they are defined. So its important to have the deny all come after all of the addresses we want to allow.

EDITOR: Fly in https://en.wikipedia.org/wiki/Classless\_Inter-Domain\_Routing

In this case, we’re saying: All IP addresses that start with 192.168.0, and ANY address that starts with 10. These are the networks that our server uses so we’ll allow requests from these networks and deny requests from any other networks.

Let’s test this configuration.

SAVE FILE

Nginx -t

Systemctl reload nginx

OPEN BROWSER

Page should load

Awesome! Our page is now only served to any requests that come from our internal network.

This works but nginx’s default 403 page isn’t very pretty. We can certainly make things look better by adding a custom 403 page.

For more information on configuring allow and deny, take a look at the documentation on nginx.org:

EDITOR: FLY IN LINK TO <http://nginx.org/en/docs/http/ngx_http_access_module.html>

# 03\_03 STARTS HERE

Along with using the allow and deny directives, we can add a custom 403 page. So even though someone might be denied from viewing a page, we’ll give them something pretty to look at.

First let’s create a location that’s always denied so we can see the default page generated by nginx:

GO BACK TO TERMINAL

vim /etc/nginx/conf.d/wisdompetmed.local.conf

I’ll add a locatio called /deny and use the deny all directive:

location /deny {

deny all;

}

Now let’s save and test this out:

SAVE FILE

Nginx -t

Systemctl reload nginx

Back in the browser

OPEN BROWSER

http://192.168.0.3/deny

Page should load

If we edit the vhost file, you might recall that we already have a configuration for handling 404 errors. Let’s just use that code over again for 403 errors.

Copy the 404 section

Change all the 404s to 403a

Previously, we used one custom page for all 500 errors. In this case, though, we won't’ use one section for all the 400 errors since “file not found” and “you are forbidden” errors are distinctly different.

SAVE FILE

Nginx -t

Systemctl reload nginx

OPEN BROWSER

Page should load

Let’s also add a location that we will always deny so we can test our custom 403 page.

location /deny {

deny all;

}

SAVE FILE

Nginx -t

Systemctl reload nginx

And now let’s make and customize the 403 page:

cp /var/www/wisdompetmed.local/404.html /var/www/wisdompetmed.local/403.html

vim /var/www/wisdompetmed.local/403.html

I’ll just change the 404 to 403 and update the message:

Change text to “We can’t show you this page”

Change title to “We can’t show it”

SAVE FILE

Nginx -t

Systemctl reload nginx

We need to reload the configuration. Now let’s try it out:

OPEN BROWSER

http://192.168.0.3/deny

Page should load

And there’s our custom 403 page. This *definitely* looks better than nginx’s default 403 page.

**Conclusion:**

Type your conclusion statement here.

**Script and Media:**

Break the script up into parts and align it with any media (slides, web, CLI, etc.)

| **Part** | **Script** | **Media** |
| --- | --- | --- |
|  |  |  |

**Exercise Files:**

**Basement:**

In the next lesson, we’ll combine our allow and deny directives with a password to make the appointments location even more secure.