2014

Hands On Labs+

Nginx and Self Signed SSL Certificates

Table of Contents

Introduction 2
Goals 2
Packages, Resources and Prerequisites 2
Document Conventions 3
General Process 4
Linux Academy Lab Server – Server Installation and Configuration 6
Nginx Installation 6
Create a Self-Signed Certificated 6
Enable SSL in the Config File 7
Test the Connection 9
Appendix A: /etc/nginx/conf.d/ssl.conf Example 10

Introduction

Security is a commonly overlooked topic of conversation when discussing Linux servers. Although it is not uncommon to talk about hardening the server itself, client access to the server can sometimes be an afterthought. Securing your Nginx web server's content can be accomplished simply by using an encrypted web session. This is accomplished by installed an SSL Certificate in Nginx and then activating the configuration.

We will talk about the security around generating SSL keys and then how to install a Self Signed Certificate can be a means to that end (and keep in mind, a certificate from a Third Party issuer is accomplished the same way).

Goals

This Hands On Lab will show you how to generate a valid Self Signed Certificate and Key for Nginx that can be installed on the local server.

Although we will be using CentOS 6.5 during the course of this document, the process is exactly the same for all Linux distributions with the exception of the location of the VHOST files on the web server. In Red Hat (RPM distributions), by default, all vhost entries are in the httpd.conf file itself and not in the 'sites-available and sites-enabled' directories as indicated here.

Packages, Resources and Prerequisites

- Nginx Web Server
- OpenSSL Server
- Text or GUI Based Web Browser

The resources you will be accessing during the course of this lab are:

- CentOS 6.5 Web Server and Your Local Browser
    - You will use this to install Nginx and the certificate on and then connect for testing from your local browser

Prerequisites to this lab:

- A LinuxAcademy.com Lab+ Subscription
- Internet Access and SSH Client
    - You will need to connect to the public IP of the server in order to configure Nginx and generate the certificate

        - SSH client can be Windows (i.e. Putty) or from another Linux system shell
    - For testing, use any browser from a system with internet access
- Login Information (Provided When The Server Starts Up)

Document Conventions

Just a couple of housekeeping items to go over so you can get the most out of this Hands On Lab without worrying about how to interpret the contents of the document.

When we are demonstrating a command that you are going to type in your shell while connected to a server, you will see a box with a dark blue background and some text, like this:

```
linuxacademy@ip-10-0-0-0:~$ sudo apt-get install package
[sudo] password for linuxacademy: <PASSWORD PROVIDED>
```

That breaks down as follows:

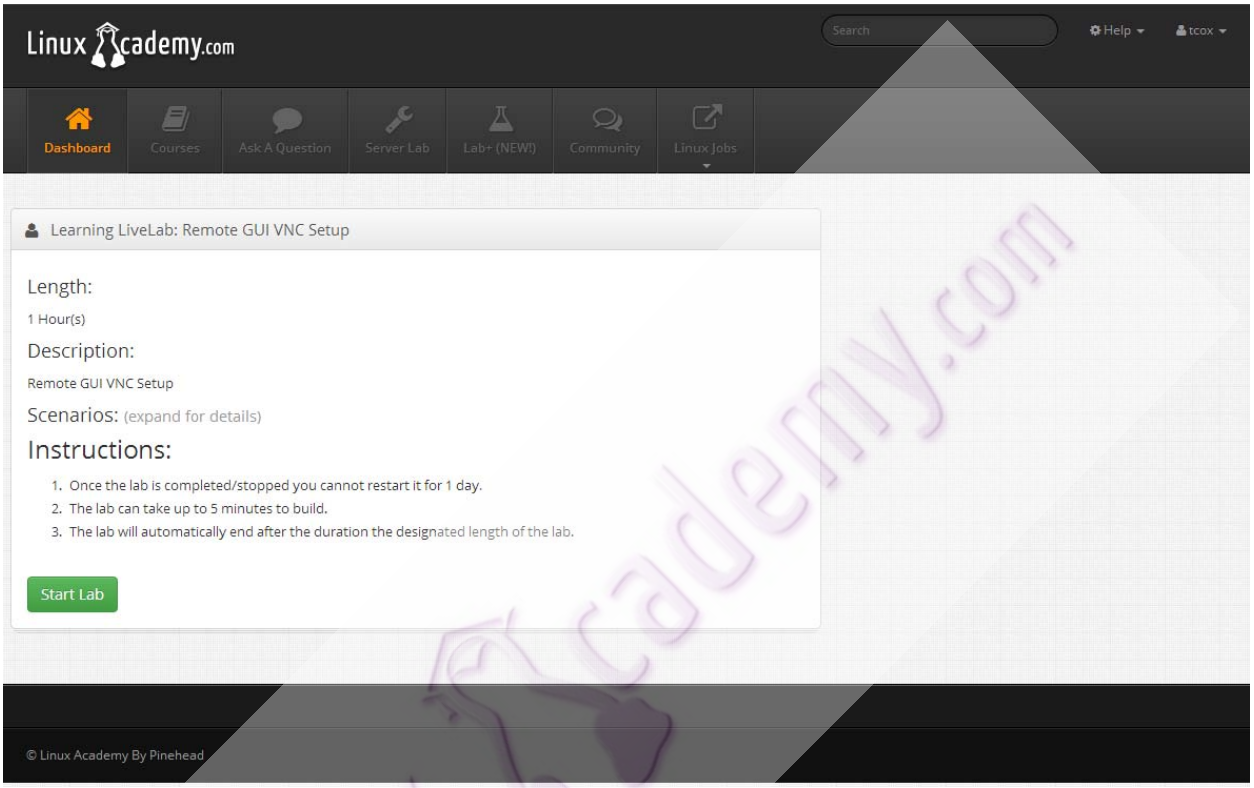
- The white text that looks something like "linuxacademy@ip-10-0-0-0:~$: "should be interpreted as the console prompt your cursor will be sitting at when you are logged into the server. You are not typing that into the console, it is just there. Note that the portion that says "ip-10-0-0-0" will be different for you based on the IP address of your system.
- The bold yellow text in any command example is the actual command you will type in your shell.
- Any lines subsequent to the bold yellow command that you type in is to be considered as the response you can expect from your command entry.

If you do not see the command prompt as the first line in the example, then all the white text is an example of a text, script or configuration file and is intended to be typed in its entirety (in the event you are instructed to create a previously non-existent file) or to compare against the contents of the file on your system.

One final convention, if you see a "~" at the end of a line in a command or text example, that will indicate that the line overflowed the end of line. Meaning you should just keep typing without hitting enter to start a new line until the natural end of the command.

General Process

When you are ready to begin the Hands On Lab, log into your Linux Academy Lab+ subscription and navigate to the "Live Labs" section on the Dashboard. Once you choose the "Nginx and Self Signed SSL Certificates" Lab from the list, you will see the screen similar to the one below:

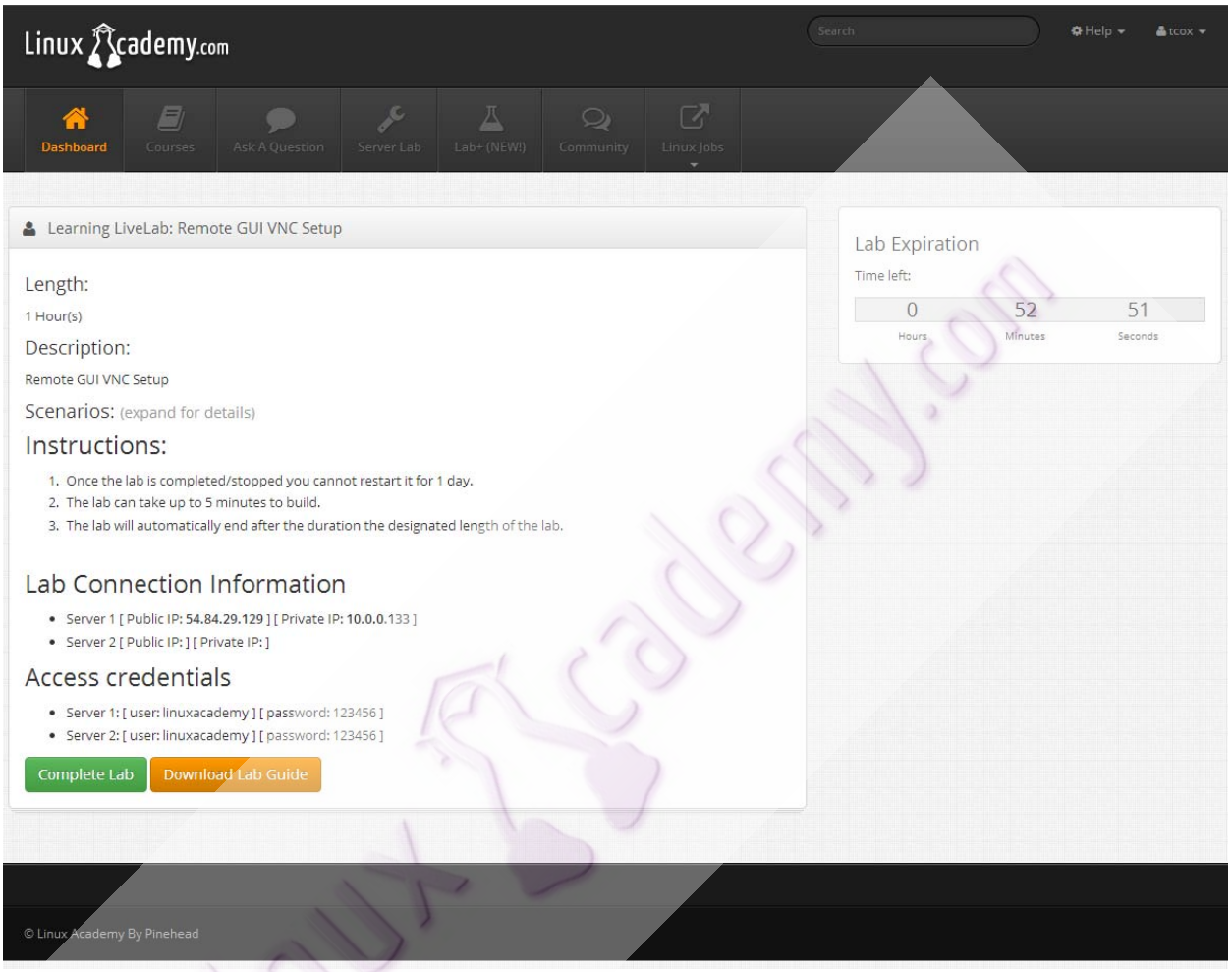


[Linux Academy Live Labs+ - Start Screen]

A few things to note before you start this process:

- When you launch the lab from this screen, it may take up to FIVE MINUTES for your servers to be deployed and be available for your use.
- Do not leave your desktop and come back, once the servers are launched, you will only have TWO HOURS to complete this lab from start to finish. After that point, the servers time out and are deleted permanently. Any and all work that you have done will then be lost.
- You can only use this lab ONCE before it will require a REST PERIOD OF ONE HOUR. If you try to use it more than that after completing the lab or the servers timing out, the screen will tell you when it will be available to you again.
- Other than those descriptions, you may retry any of the Labs+ labs as many times as you wish as long as you are a subscriber.

Once you have clicked on the 'Start Lab' button that you see above, a process will launch on our servers that will deploy the two servers we will use in our lab for testing. After a few minutes of processing (and you will see a status message that says "Creating Lab… Please Wait"), you should see a screen that looks like this one:



[Linux Academy Live Labs+ - Start Screen]

You will see all the information you need to access your servers from another system. Specifically, you need:

- The server public IP addresse
- Access credentials

One thing to note is that, in addition to the IP that you see above, the server will have another IP assigned to it in the 10.0.0.x subnet. This is a private IP address and will not route outside of your private server pool. Your server will have a static private address of 10.0.0.100. We will be

using the external IP address to connect over SSH as well as when configuring Nginx and our certificates.

Linux Academy Lab Server – Server Installation and Configuration

Connect to your Linux Academy Lab server over SSH using any Windows or Linux SSH client or shell. We will generate all of our configuration and certificates from the command line while connected to the CentOS 6.5 Server in this lab.

Nginx Installation

Once we are logged into our Linux Academy Lab Server, we need to install a couple of things in order to generate, install and test our secure server. Open a command prompt and type in the following in order to install the repository we need for Nginx:

```
sudo wget http://download.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
rpm -ivh epel-release-6-8.noarch.rpm
```

This will install all the necessary entries for the Epel 3rd Party Repository. Since Nginx is not part of the core CentOS package tree, we need this repository in order to avoid downloading and compiling it from scratch (something you can do, but is outside the scope of this lab). Once this is done, you can run 'sudo yum update' and then the following in order to install what we need for the rest of our lab:

```
sudo yum install telnet openssl nginx
```

We now have the web server installed, telnet for our testing later on as well as the openssl client that will allow us to create our SSL certificates for installation (whether we were to need to generate keys for a self signed certificate, as in our lab today, or a key to obtain a signed certificate from a third part like Entrust, the process of generating and install the keys locally are the same).

Create a Self-Signed Certificated

Now that we have Nginx installed along with the SSL modules and support libraries we need, let's generate the keys and certificate files needed to secure our website traffic. First, let's create a directory for our local SSL keys and certificate files.

```
sudo mkdir /etc/nginx/ssl
```

After we have a directory for our certificates, let's go ahead and create what we need:

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 ~
-keyout /etc/nginx/ssl/nginx.key -out ~ /etc/nginx/ssl/nginx.crt
```

After you execute this command, you will be asked to enter a bunch of information that will be incorporated into your key and your certificate. Since this is a self-signed certificate that will show a browser warning in any event (since you are not considered a trusted certificate issuing authority), what you enter is largely unimportant. If we were generating just the key file to send to a valid issuer, it would. You will see something like the following:

```
Generating a 2048 bit RSA private key
.....................+++
.....................................................................
.....................................................+++
writing new private key to 'nginx.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a
DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:WY
Locality Name (eg, city) []:Lourdes
Organization Name (eg, company) [Internet Widgits Pty Ltd]:My Name Company
Organizational Unit Name (eg, section) []:Information Technology
Common Name (e.g. server FQDN or YOUR name) []:ip-10-0-0-
100.linuxacademy.com
Email Address []:admin@linuxacademy.com
```

At this point, our certificate key and the certificate have been written to the previously created '/etc/nginx/ssl' directory.

Enable SSL in the Config File

Now we have to enable the SSL configuration in our server by uncommenting the appropriate section in our Nginx configuration file. Edit the file '/etc/nginx/conf.d/ssl.conf' and be sure to find and then uncomment the following section:

```
# HTTPS server
server {
    listen       443;
    server_name example.com;
```

```
    ssl on;
    ssl_certificate /etc/nginx/ssl/nginx.crt;
    ssl_certificate_key /etc/nginx/ssl/nginx.key;
}
```

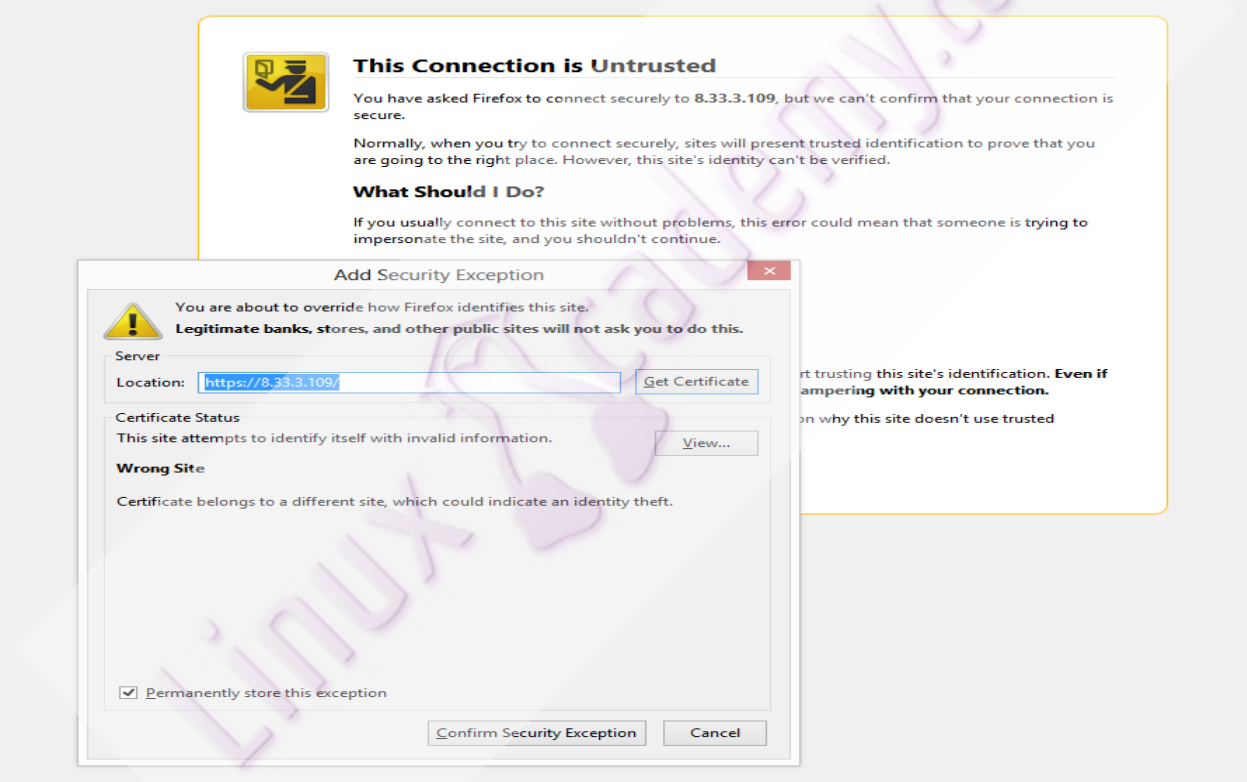
Keep in mind that some of the output may differ slightly for your installation depending on whether you are using the Linux Academy Servers or your own distribution. As long as you don't receive an error message that the SSL Module doesn't exist, you are ready to start the web service and verify everything is working the way we expect.

Test the Connection

Finally, all we need to do is open a browser and navigate to the external IP address of the site and confirm it comes up.

NOTE: This is a self-signed certificate and, although the traffic between your client browser and the Nginx server is encrypted via SSL, your browser will warn you that the connection is potentially insecure. That is because it was not issued or verified by a trusted certificate authority (like Verisign or Entrust). We are issuing a self-signed certificate as a matter of demonstration and because it is generated at no cost. A certificate authority can issue verified certificates, but they come at a cost from $49 to more than $3500 depending on the encryption type and verification level (more expensive for securing credit card ecommerce transactions).

Your browser warning will look something like this:



However, you will then know that your system is indeed answering on port 443 externally, using your certificate. You can view the certificate details to confirm. You have now set up SSL and tested your configuration!

Appendix A: /etc/nginx/conf.d/ssl.conf Example

```
#
# HTTPS server configuration
#

server {
    listen       443;
    server_name  _;

    ssl                  on;
    ssl_certificate      /etc/nginx/ssl/nginx.crt;
    ssl_certificate_key  /etc/nginx/ssl/nginx.key;

    ssl_session_timeout  5m;

    ssl_protocols  SSLv2 SSLv3 TLSv1;
    ssl_ciphers  ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP;
    ssl_prefer_server_ciphers   on;

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