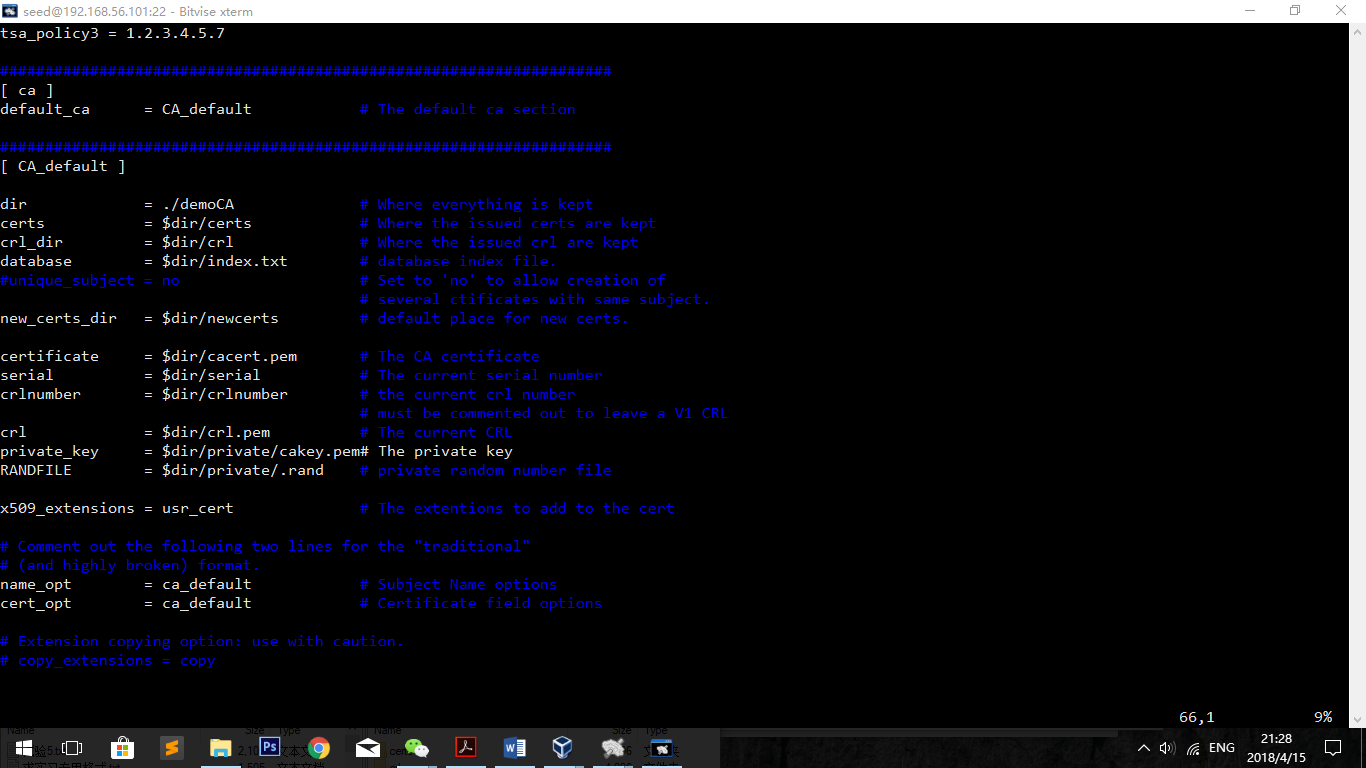
PKI Lab Report

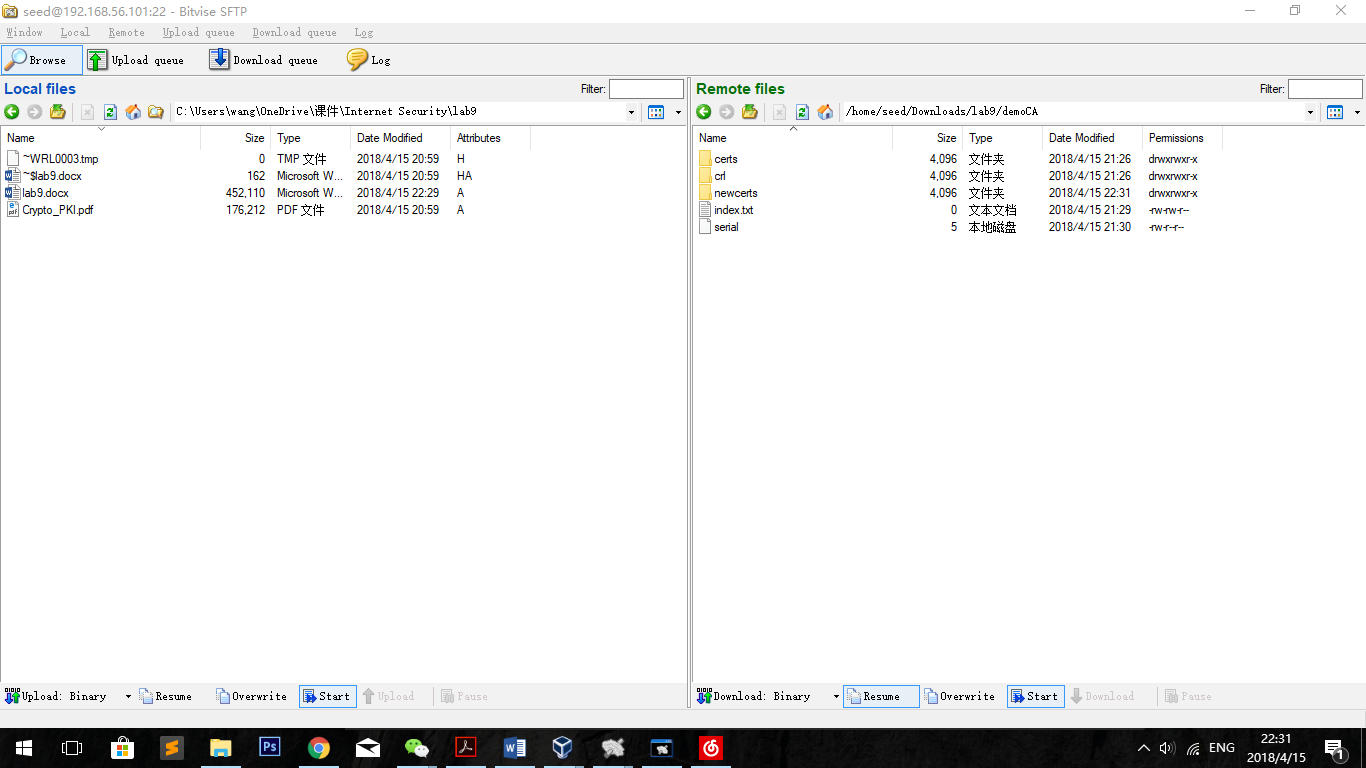
## 2.1 Task 1: Becoming a Certificate Authority (CA)

### Get the file from local machine.

sudo cp /usr/lib/ssl/openssl.cnf /home/seed/Downloads/lab9/

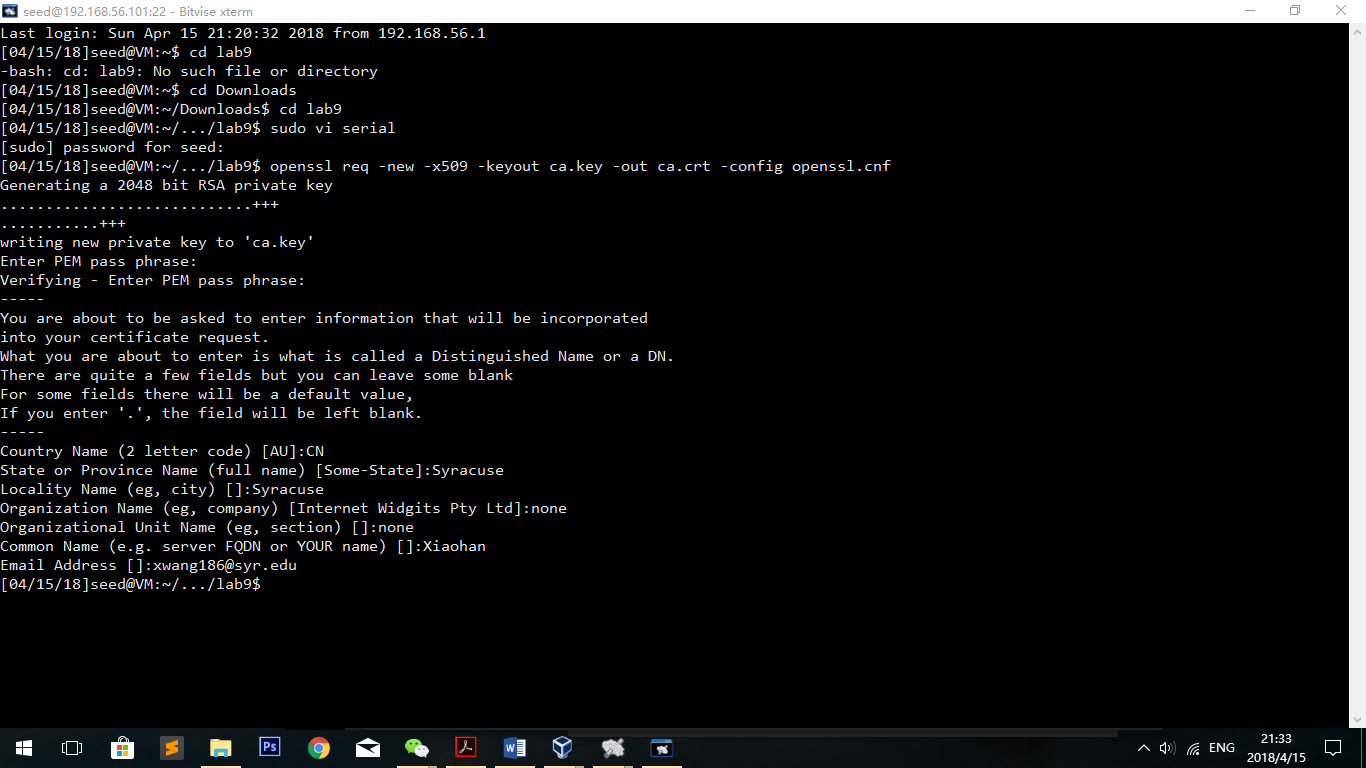
### Build dictionaries according to the cnf file:





### Generate crt files

openssl req -new -x509 -keyout ca.key -out ca.crt -config openssl.cnf

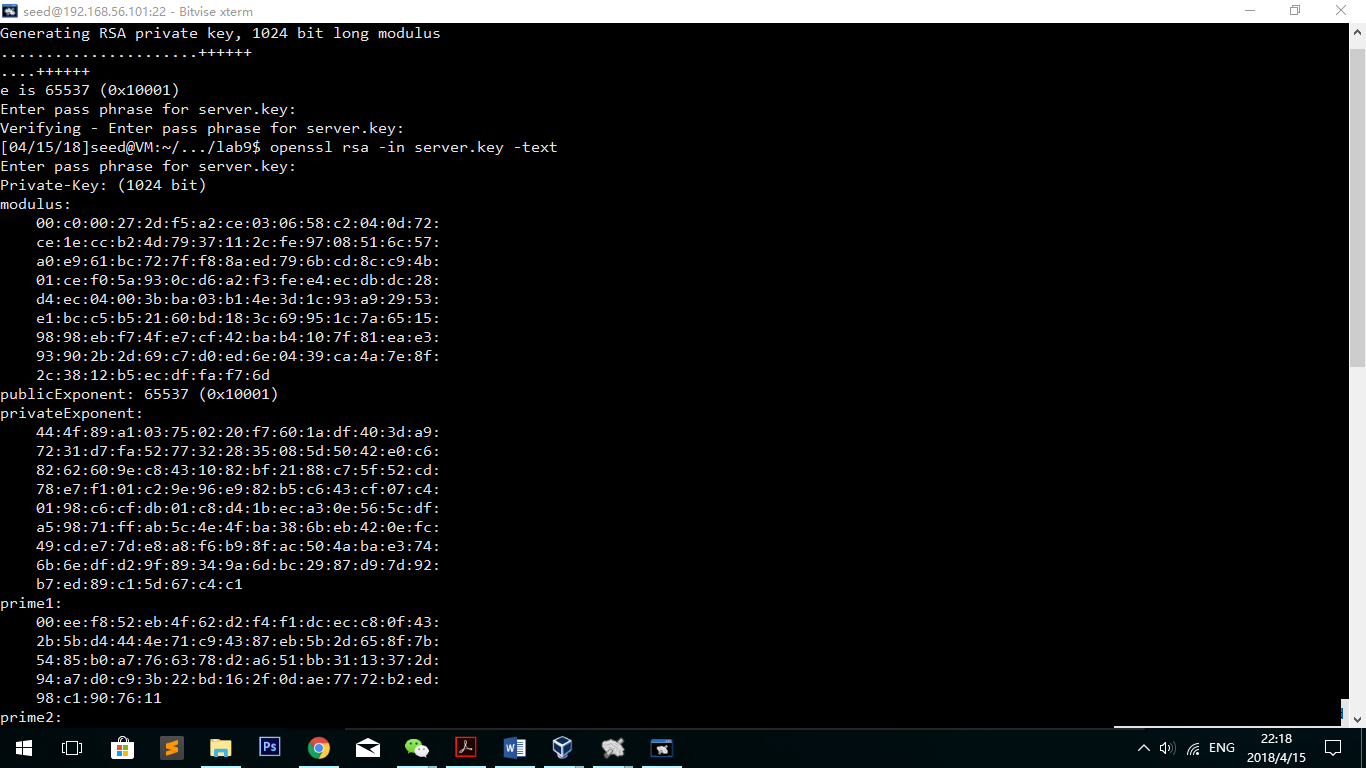


## 2.2 Task 2: Creating a Certificate for SEEDPKILab2018.com

### Generate public/private key pair.

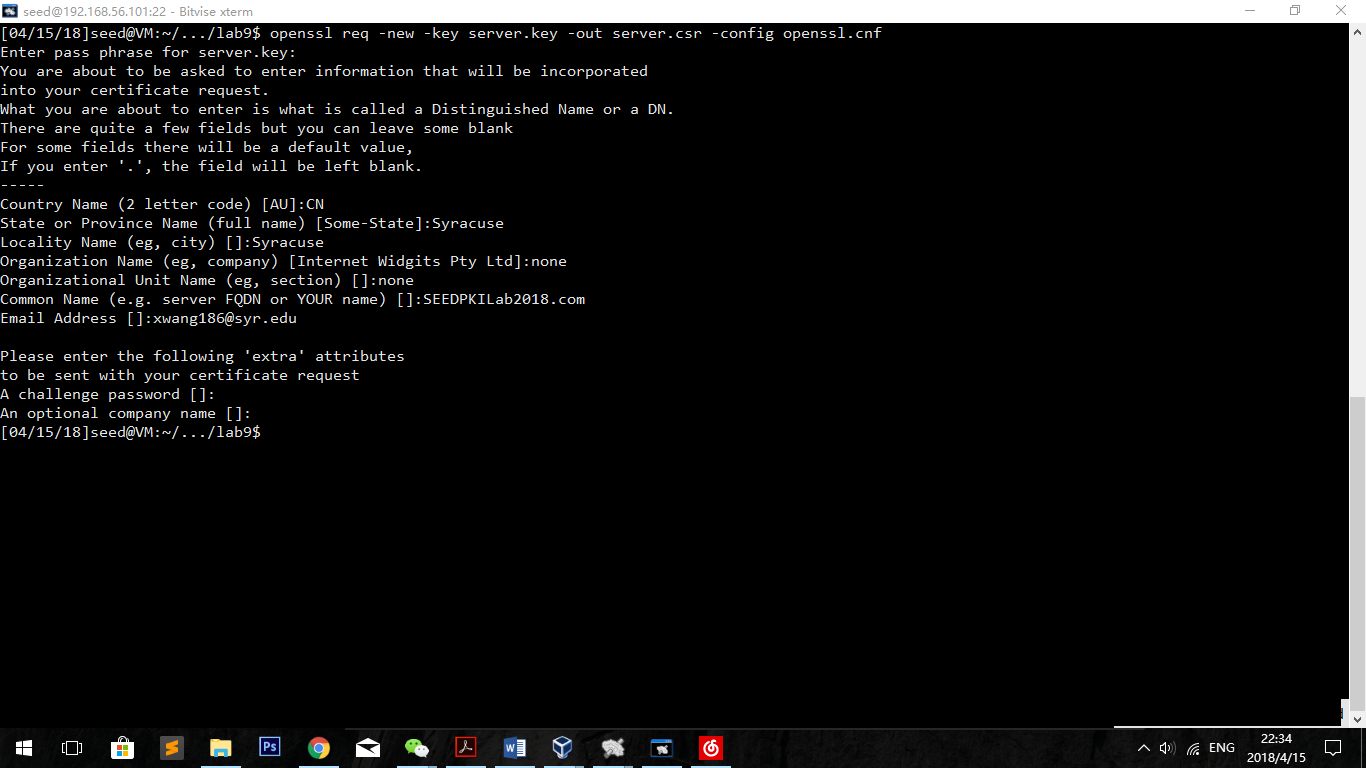
openssl genrsa -aes128 -out server.key 1024

openssl rsa -in server.key -text



### Generate a Certificate Signing Request (CSR).

openssl req -new -key server.key -out server.csr -config openssl.cnf



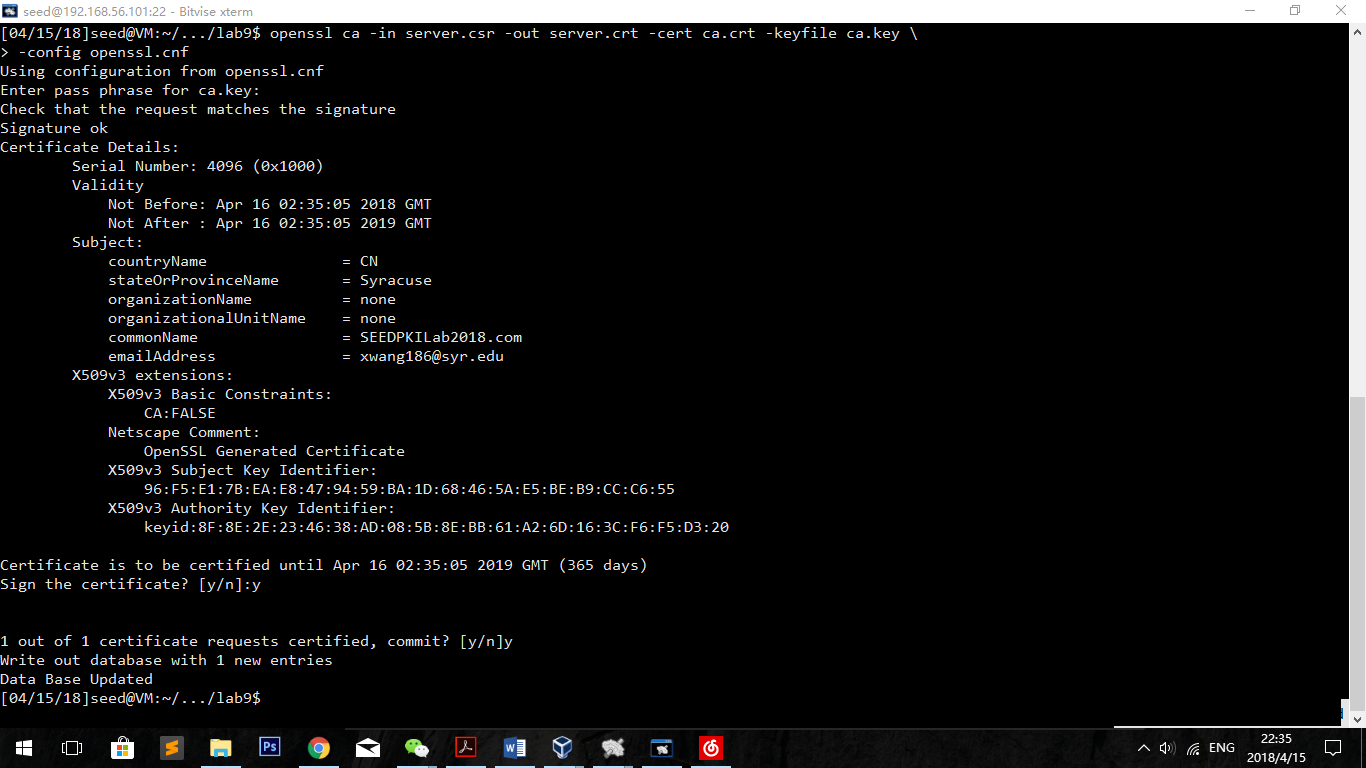
**Observation:**

Except the common name part, it is the same as task 1. In the common name field, we type in SEEDPKILab2018.com as the common name of the certification request.

### Generating Certificates.

openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key \

-config openssl.cnf



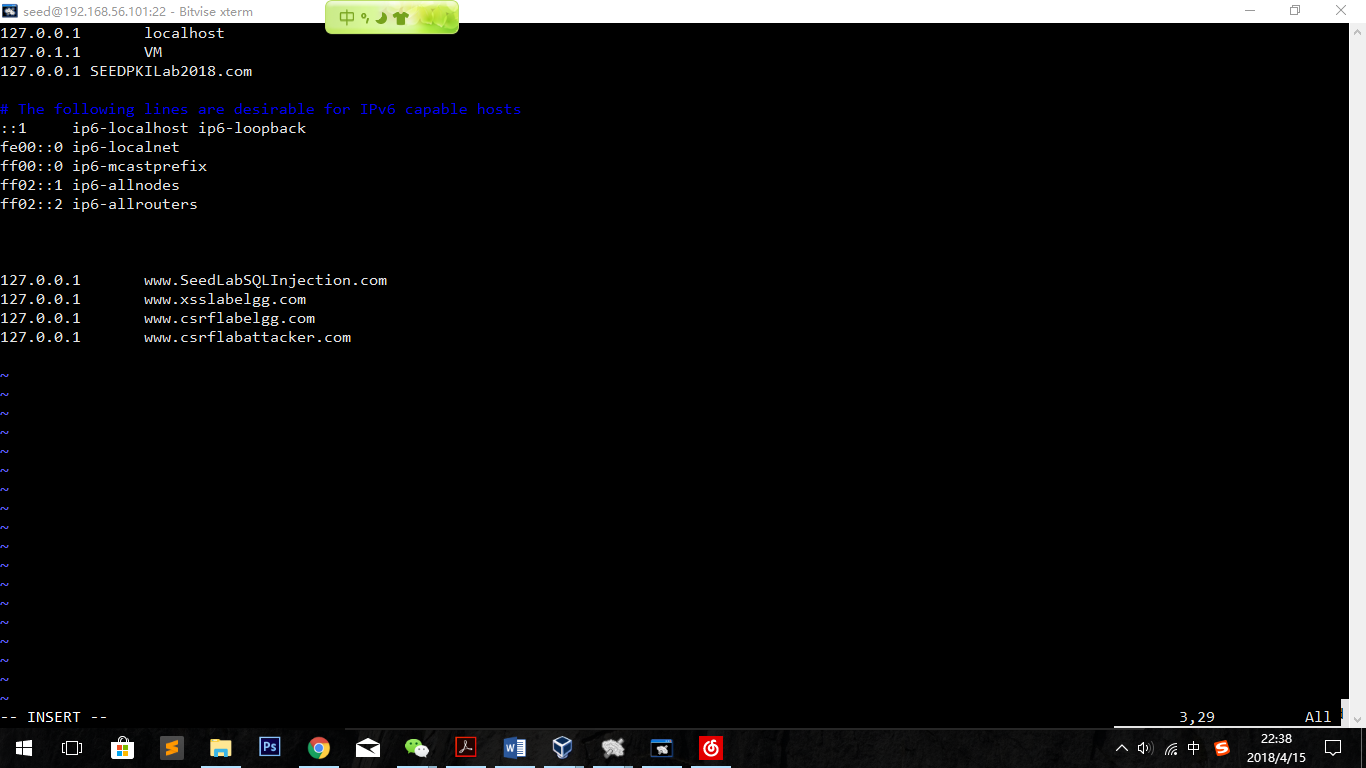
**Observation:**

The signature is signed successfully.

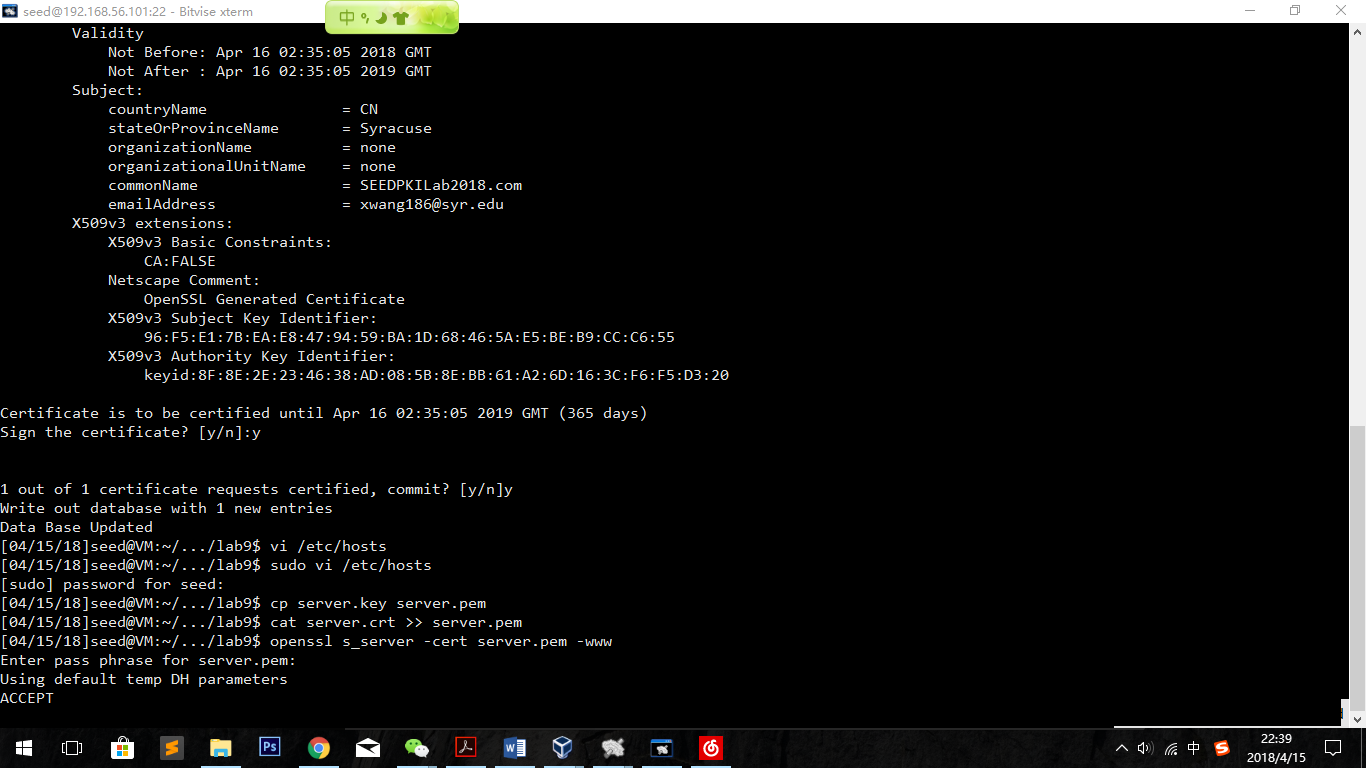
## Task 3: Deploying Certificate in an HTTPS Web Server

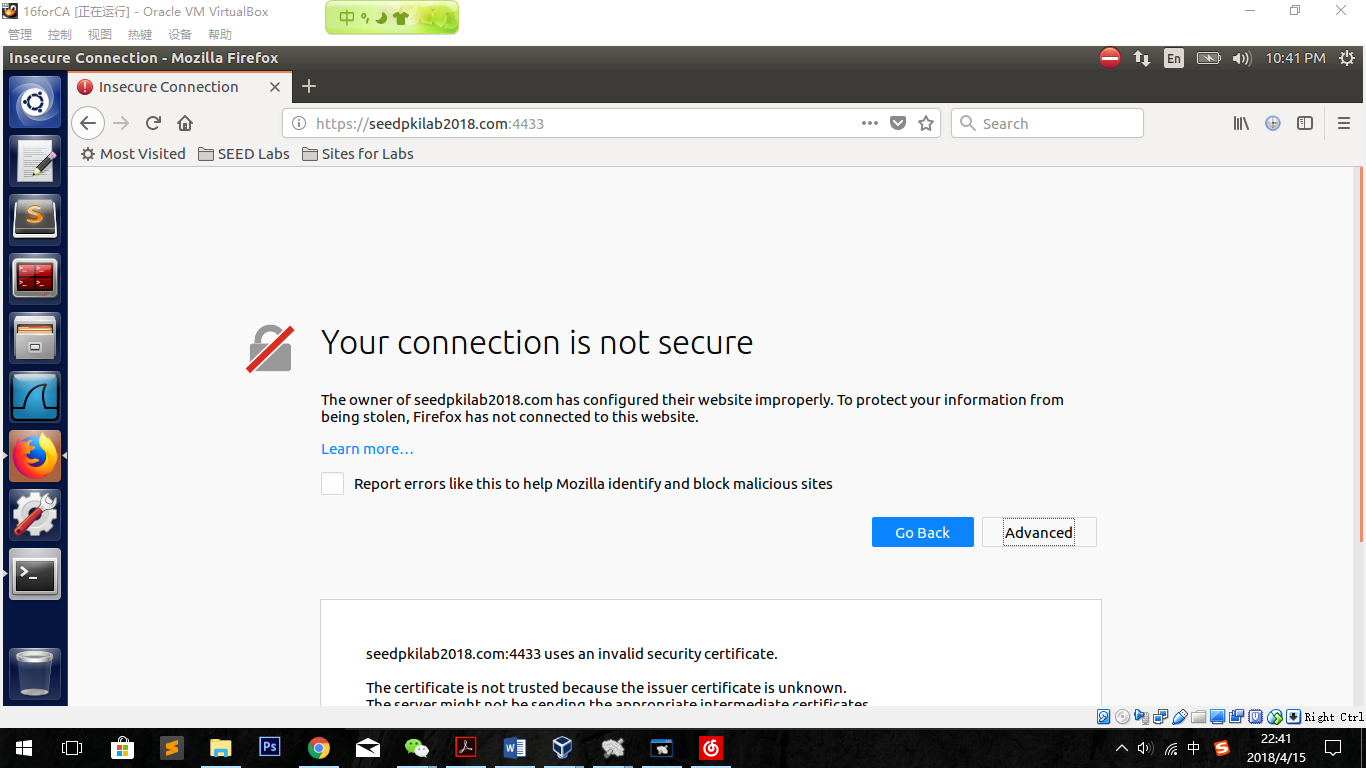
### Configuring DNS.

Edit host file

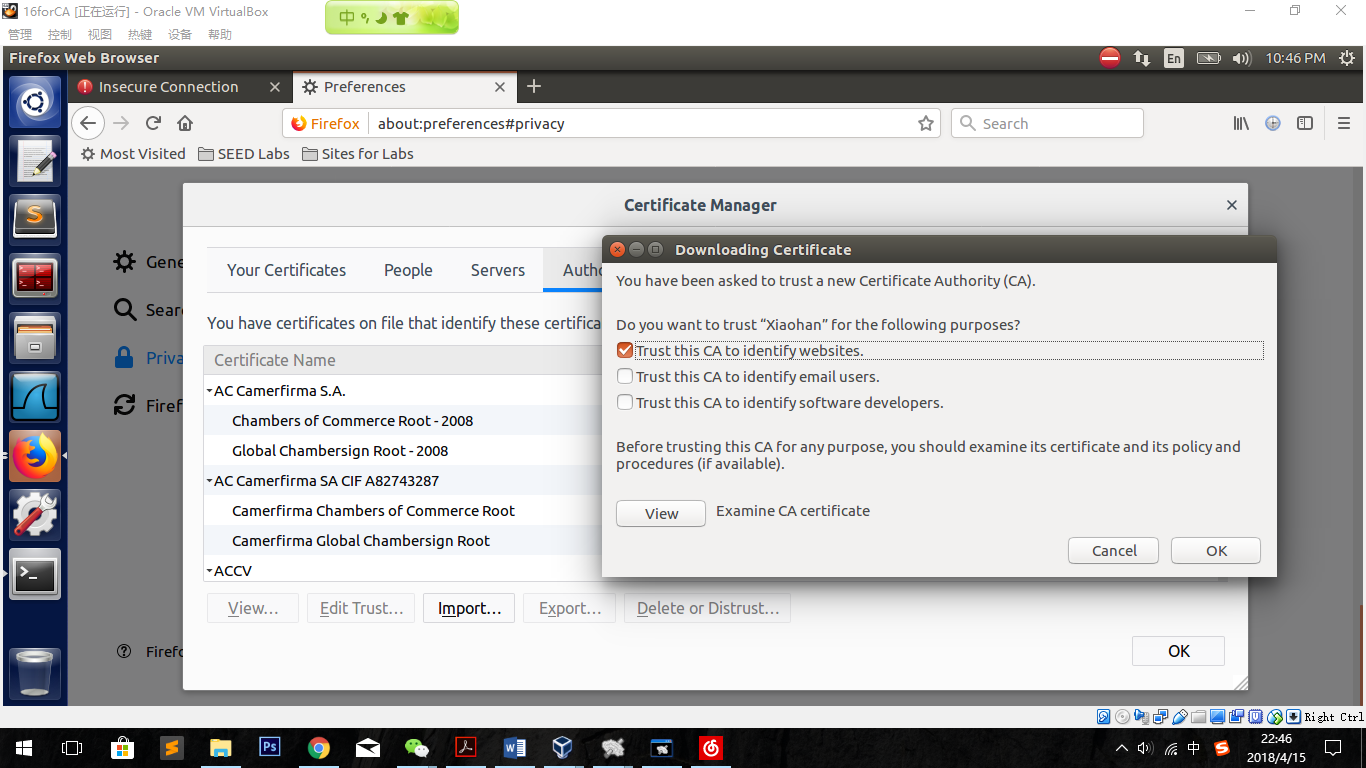


### Configuring the web server.



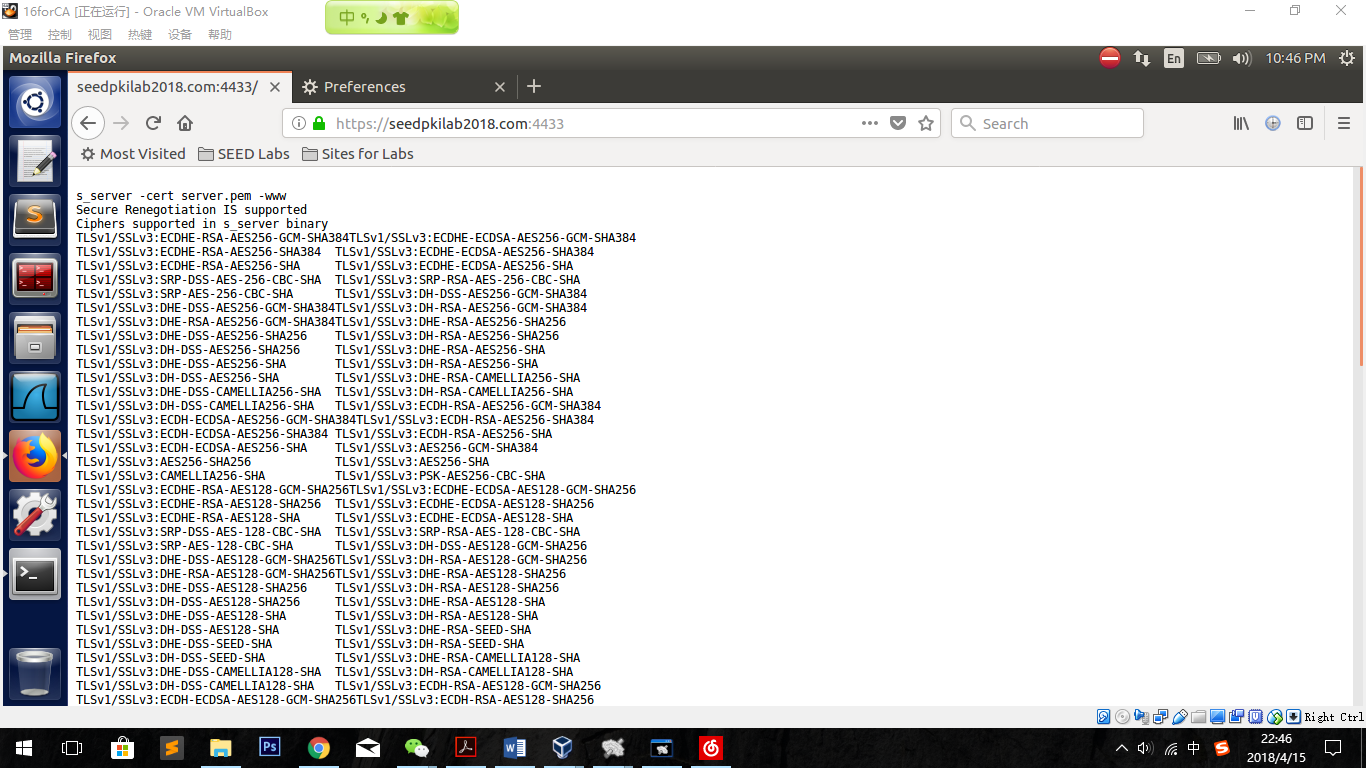


### Getting the browser to accept our CA certificate.



### Testing our HTTPS website.

#### 1.The original website



**Observation:**

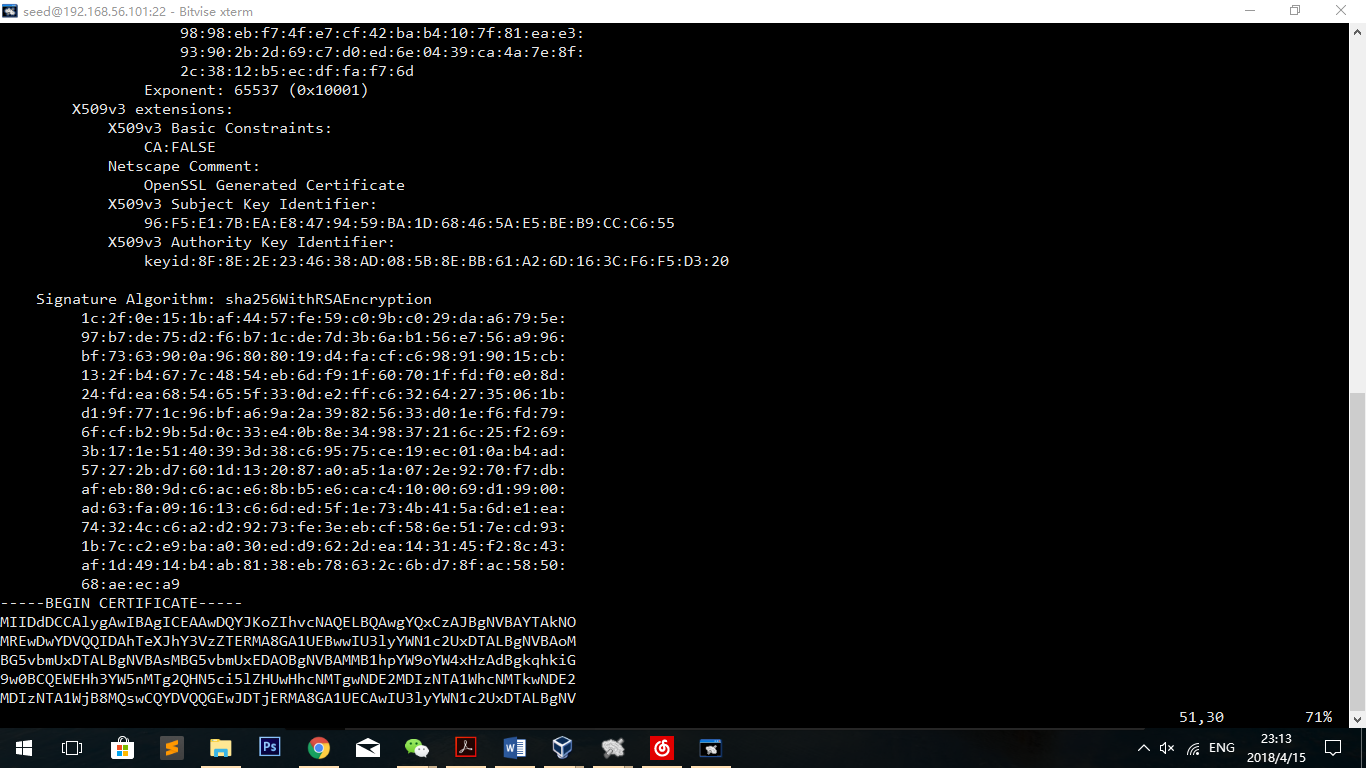
The website is successfully working. All the ciphers are shown in the website.

**Explanation:**

Since the ca.crt file is imported to the browser, while visiting the website the browser will first check the CA information. In the ca.crt file, the website is successfully signed by the root CA. That is why the website is now working.

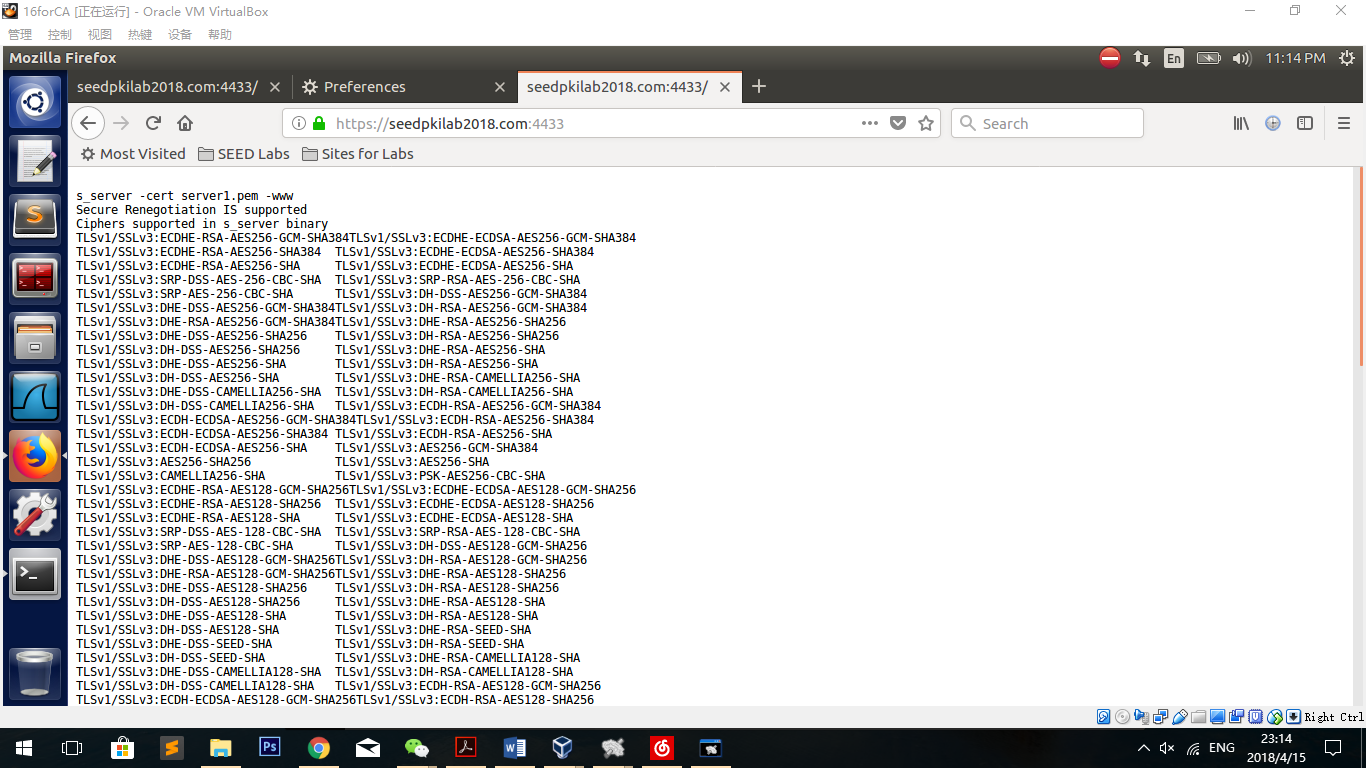
#### 2. Modify a single byte of server.pem, and restart the server

Modify:



Change the beginning of key id from 8F to 9F

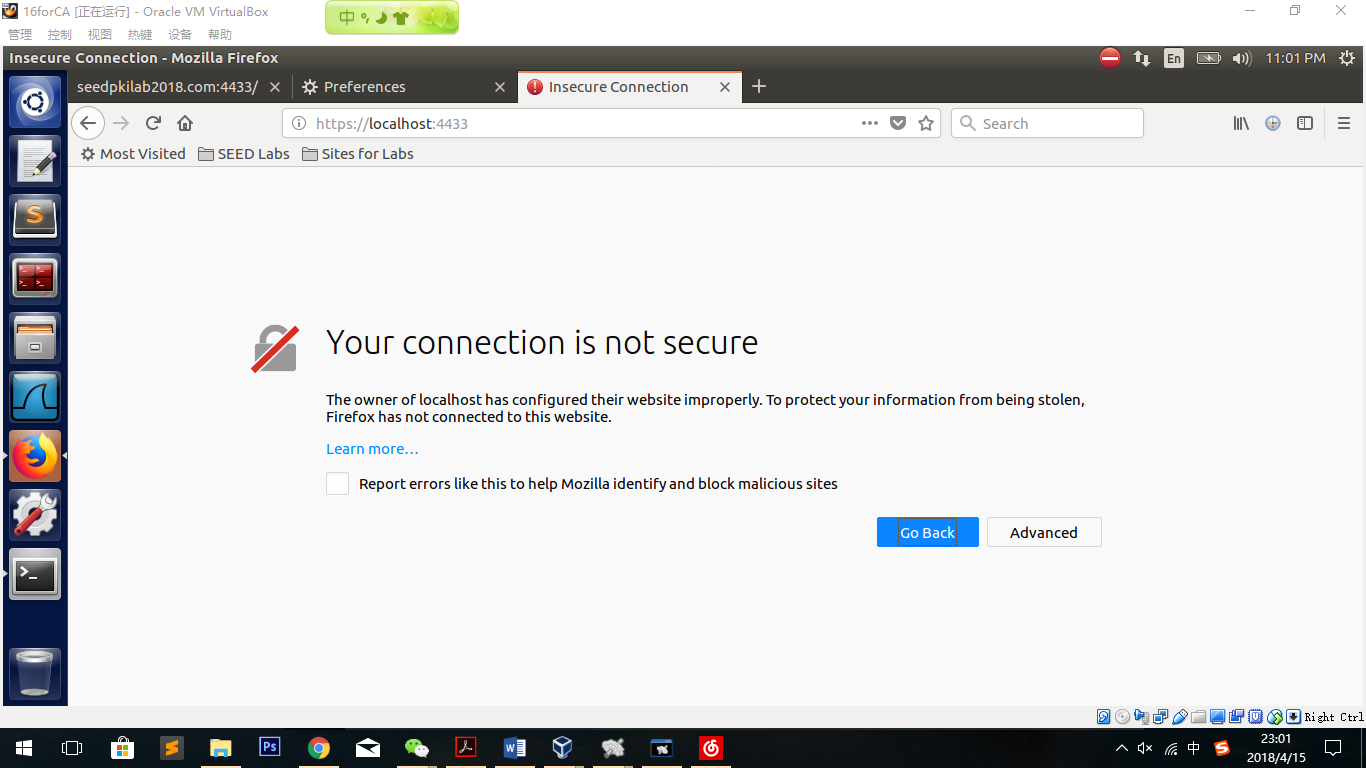




After changing the key\_id, we can still visit the website successfully.

So, we can draw the conclusion that changing the field of “X509v3 Authority Key Identifier:” doesn’t effect the result.

#### 3.use <https://localhost:4433> instead



**Observation:**

It shows the same result as we test before importing the CA file. The website shows that the website is not secure.

**Explanation:**

In the certification process, the domain name is also the parameter of the certification. But while visiting the website of localhost, the domain name is different from what in the ca file. In the ca file, only seedpkilab2018.com is certified. That is why localhost cannot pass the certification.

## Task 4: Deploying Certificate in an Apache-Based HTTPS Website

### Configure

Create a folder and an index page:

szudo mkdir /var/www/pkilab

sudo cp /var/www/html/index.html /var/www/pkilab

sudo vi /etc/apache2/sites-available/000-default.conf

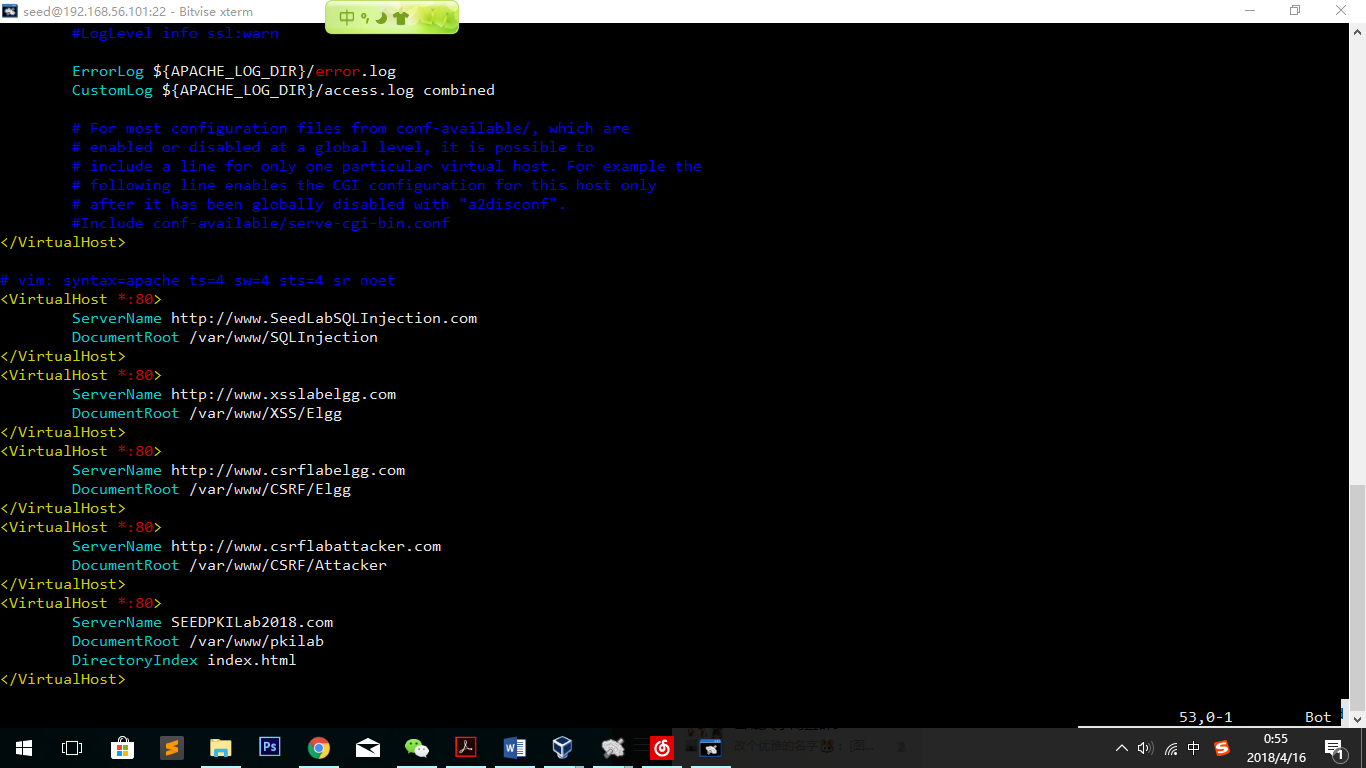
<VirtualHost \*:80>

ServerName SEEDPKILab2018.com

DocumentRoot /var/www/pkilab

DirectoryIndex index.html

</VirtualHost>



sudo vi /etc/apache2/sites-available/default-ssl.conf

<VirtualHost \*:443>

ServerName SEEDPKILab2018.com

DocumentRoot /var/www/pkilab

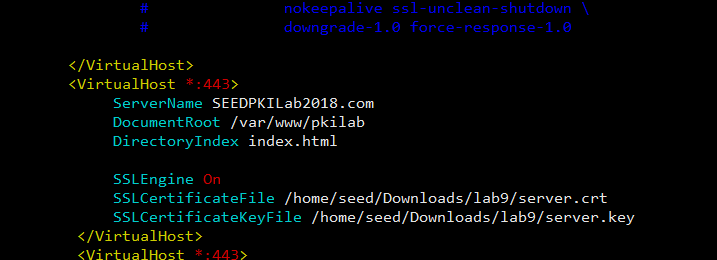
DirectoryIndex index.html

SSLEngine On

SSLCertificateFile /home/seed/Downloads/lab9/server.crt

SSLCertificateKeyFile /home/seed/Downloads/lab9/server.key

</VirtualHost>



### Test, reload and restart:

sudo apachectl configtest

sudo a2enmod ssl

sudo a2ensite default-ssl

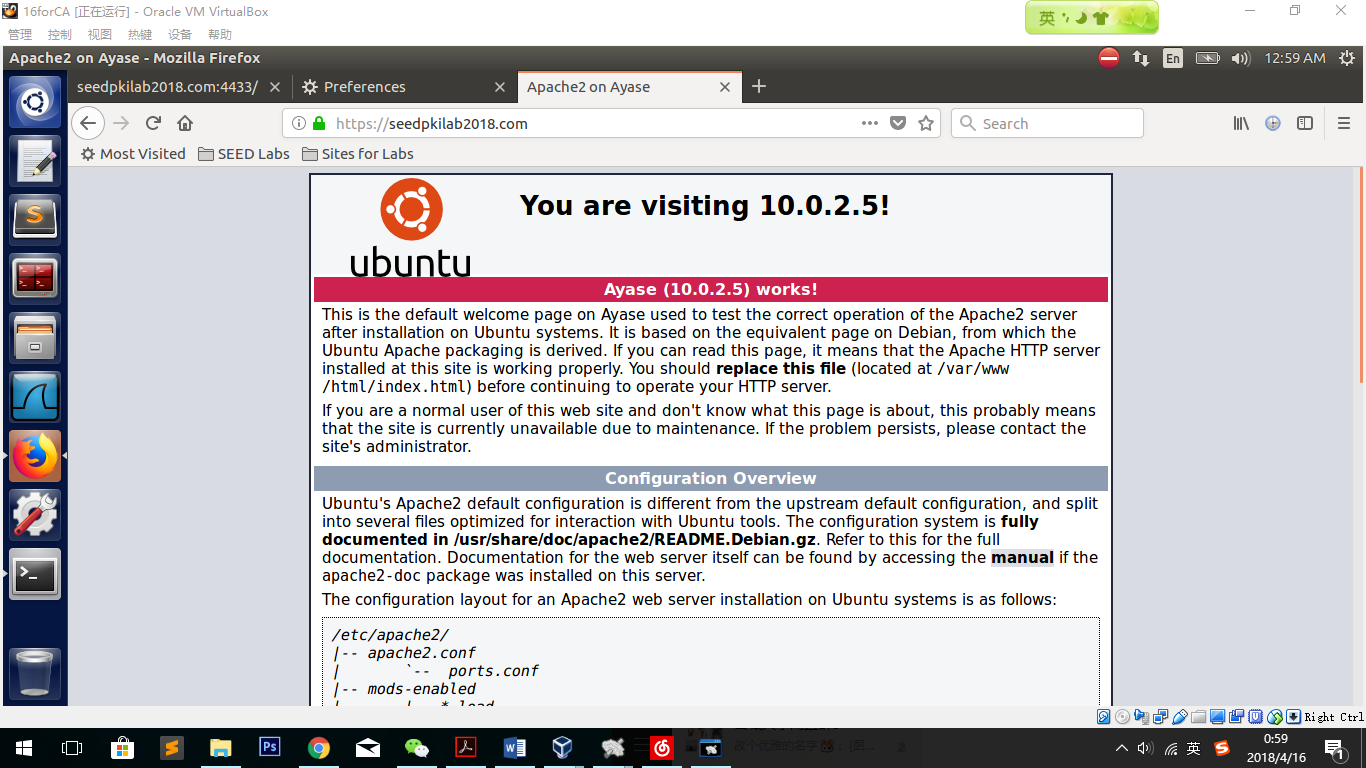
sudo service apache2 restart



**Observation:**

Set up clear! We can do test now.

### Test in the browser:



**Observation:**

After typing in the web address, the index page we copied from html folder appears. It seems that we have successfully visited the index page using the new https host.

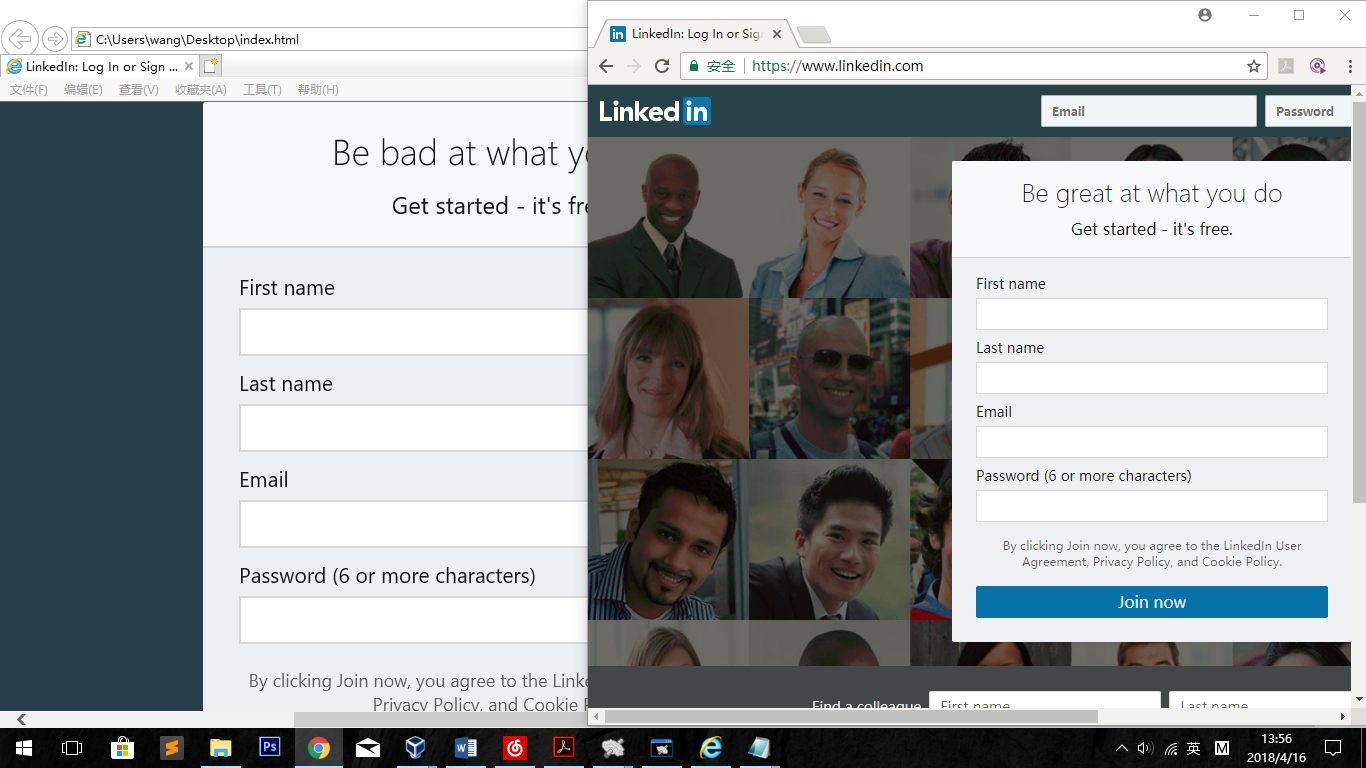
## Task 5: Launching a Man-In-The-Middle Attack

### Step 1: Setting up the malicious website.

Repeat task5 with the server name of

https://www.linkedin.com/

Firstly, download the resource code of the website linkedin and change something to show that what we want to use is a fake one.



I changed “great” to “bad” in the website. Right one is the original Linedin and the left one is the fake one. They look similar and if I successfully let the fake website working as the man in the middle then I succeed.

sudo mkdir /var/www/linkedin

sudo cp /home/seed/Downloads/lab9/index.html /var/www/linkedin

sudo vi /etc/apache2/sites-available/000-default.conf

<VirtualHost \*:80>

ServerName www.linkedin.com

DocumentRoot /var/www/linkedin

DirectoryIndex index.html

</VirtualHost>

sudo vi /etc/apache2/sites-available/default-ssl.conf

<VirtualHost \*:443>

ServerName www.linkedin.com

DocumentRoot /var/www/linkedin

DirectoryIndex index.html

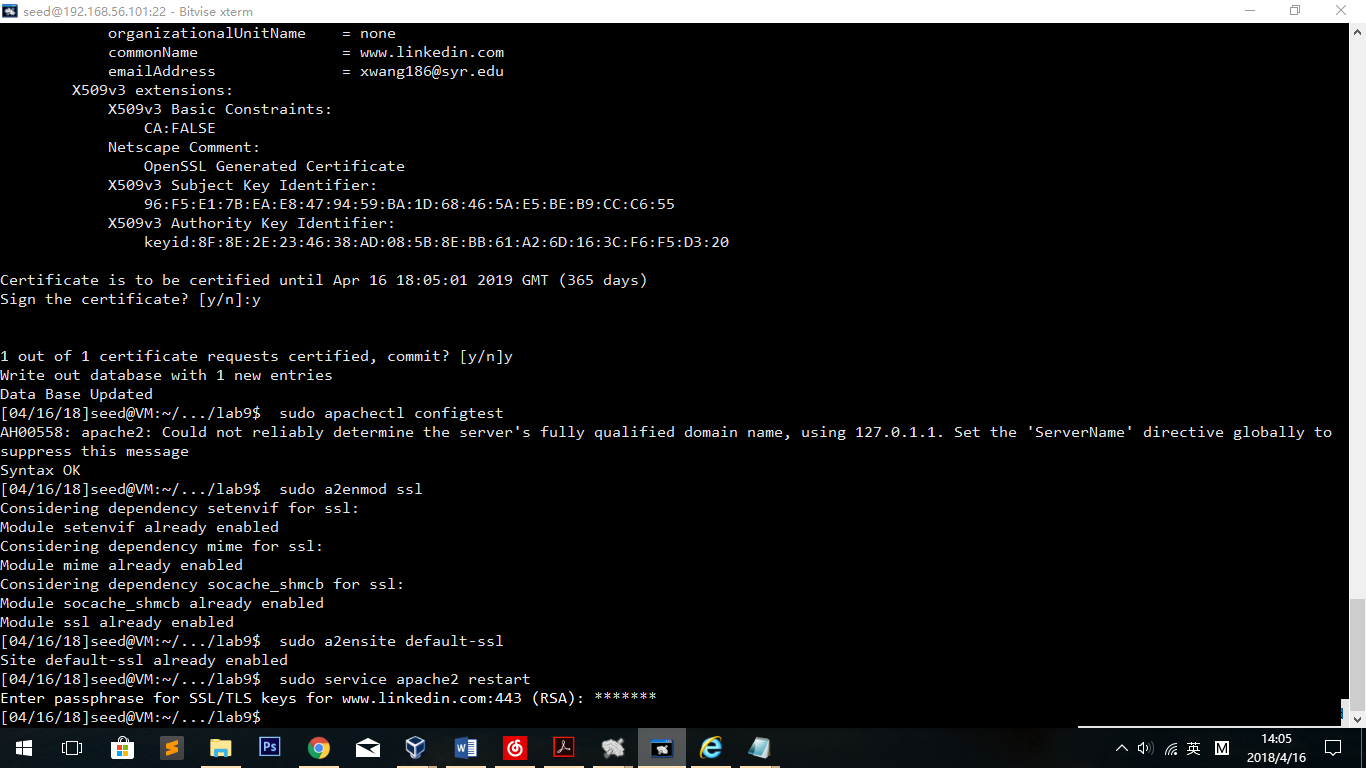
SSLEngine On

SSLCertificateFile /home/seed/Downloads/lab9/server.crt

SSLCertificateKeyFile /home/seed/Downloads/lab9/server.key

</VirtualHost>

Test reload and restart:



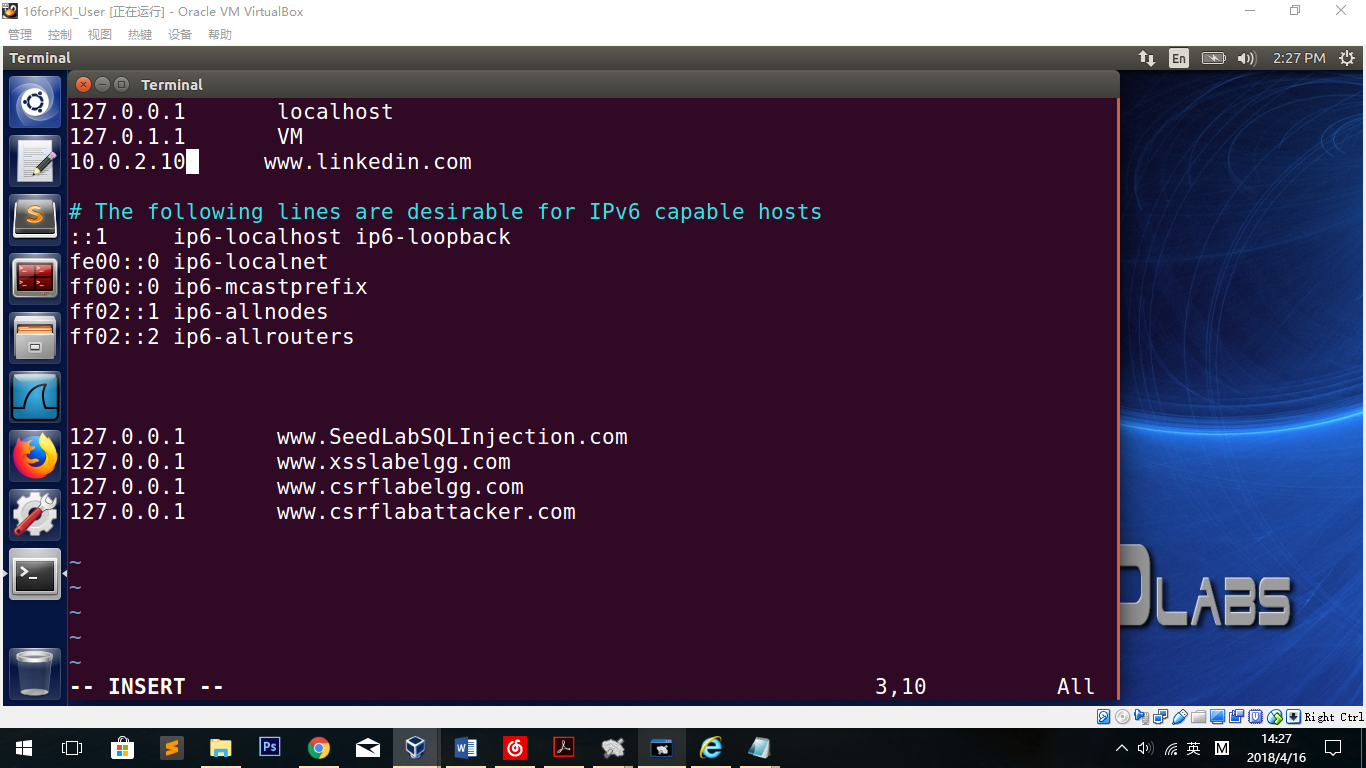
Observation: The new page is successfully set.

### Step 2: Becoming the man in the middle

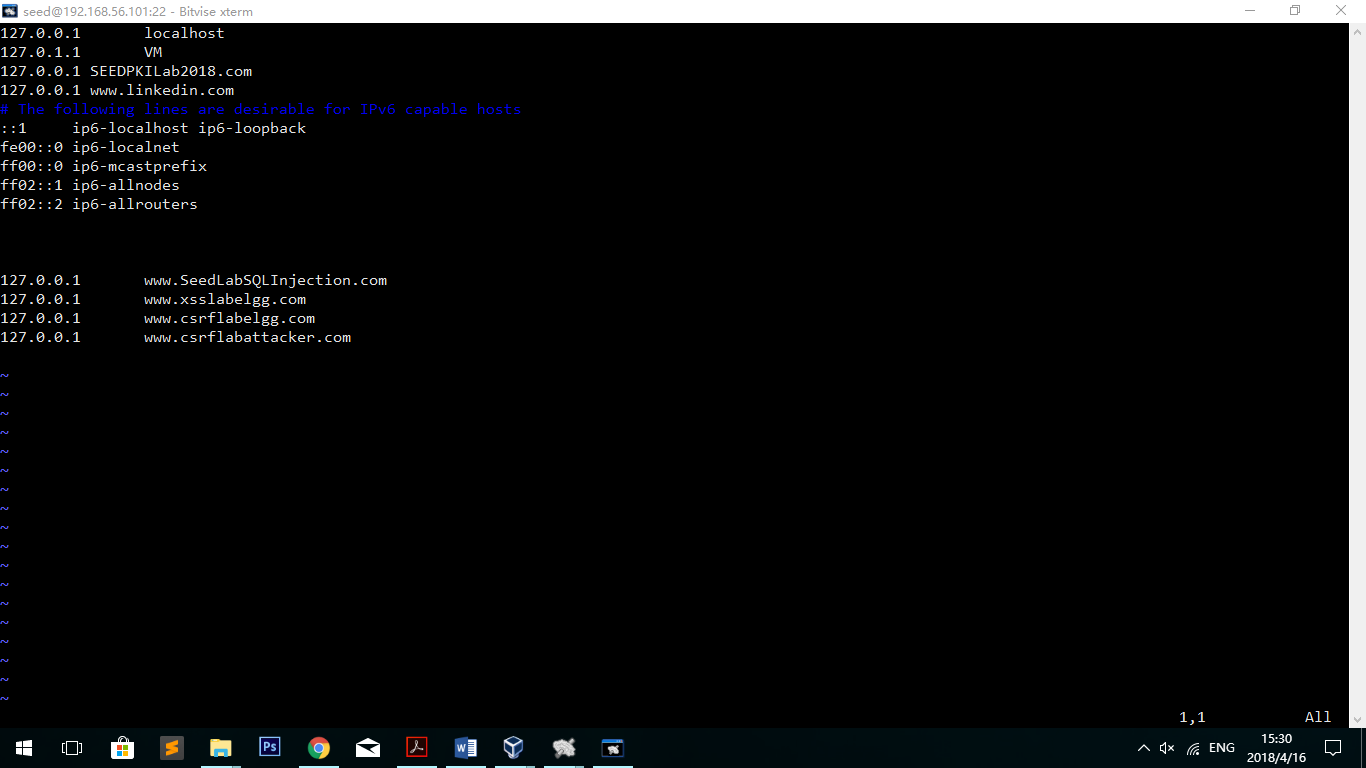
Before we do this, import the ca in the user’s machine.

Build a user machine using NAT network. (Machine name 16forPKIuser)

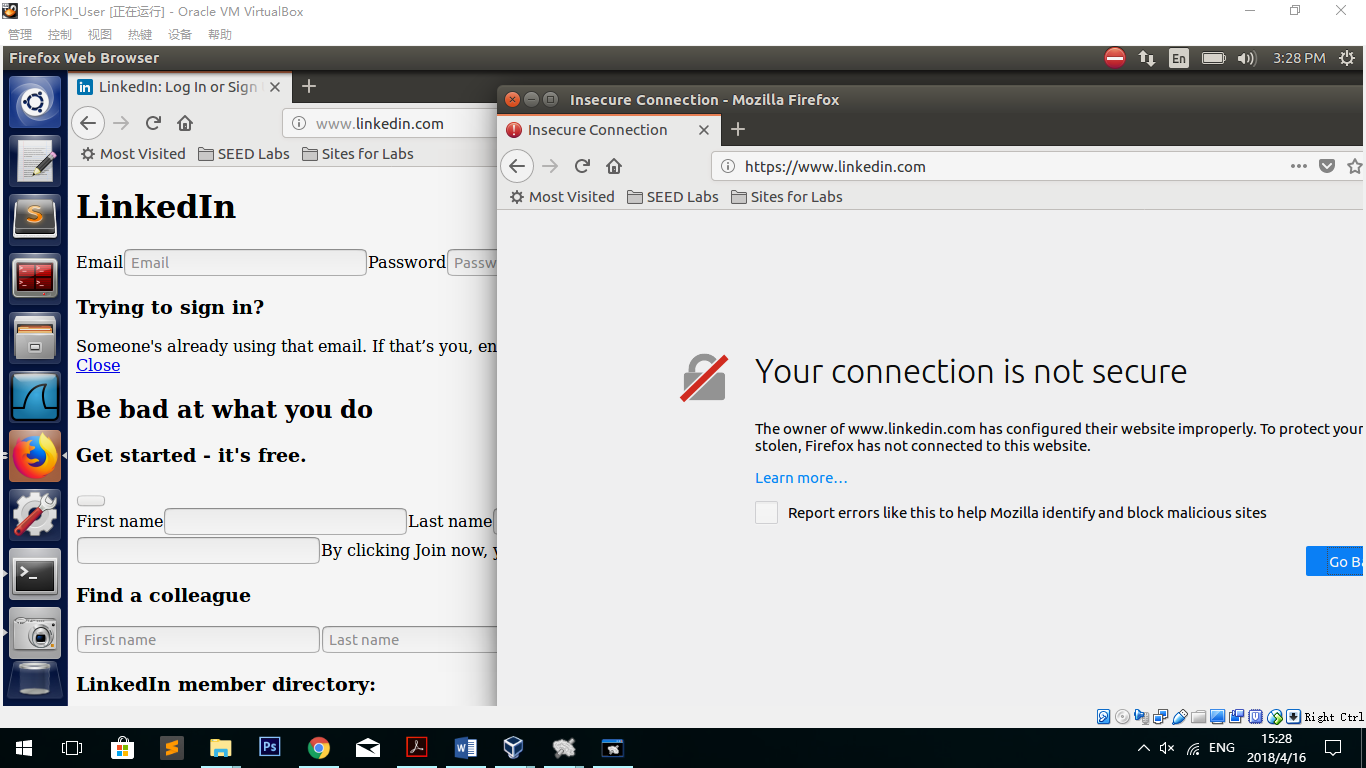
10.0.2.10 is the ip address of the malicious server (Machine name: 16 for CA)



To compare, we change the malicious machine as well.



### Step 3: Browse the target website.



**Observation**:

The HTTP attack is successful. But while testing the https, the result shows that the website is considered insecure by the browser.

**Explanation**:

In this part, the certification we used is the one when we set the seedpkilab2018.com. As a result, while checking the domain name, the new domain name [www.linkedin.com](http://www.linkedin.com) will not be certified to pass the certification.

## Task 6: Launching a Man-In-The-Middle Attack with a Compromised CA

### Experiment Design:

To show that with the private key of ca we can do attacks on any https website, we need to use the private key to sign one more time using the website as the domain name. After signing the website, we can use the new certification file in the configure.

As well, we will use linkedin.com as the example.

### Do the experiment:

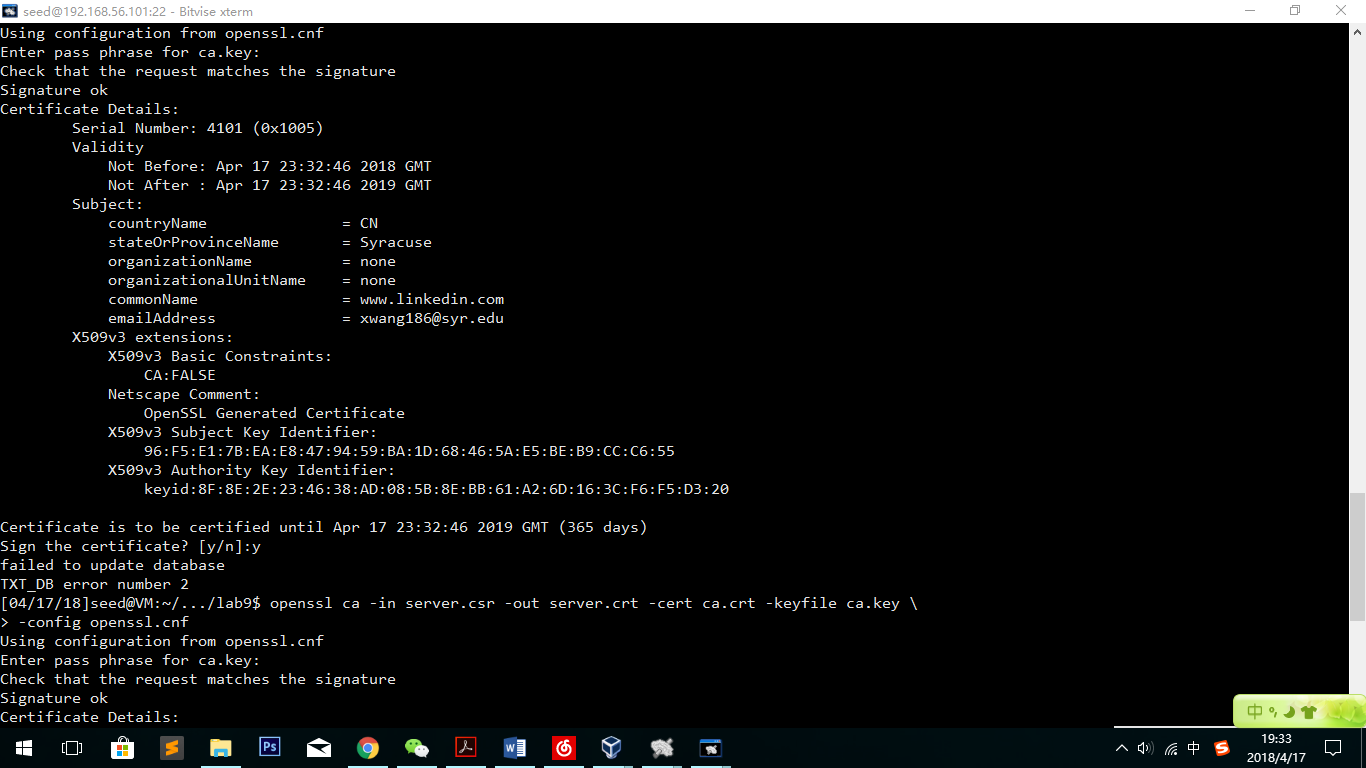
#### Make a certification of [www.linkedin.com](http://www.linkedin.com).

(The new file will replace the old file of seedpkilab2018.com)

openssl req -new -key server.key -out server.csr -config openssl.cnf

sudo openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key \

-config openssl.cnf



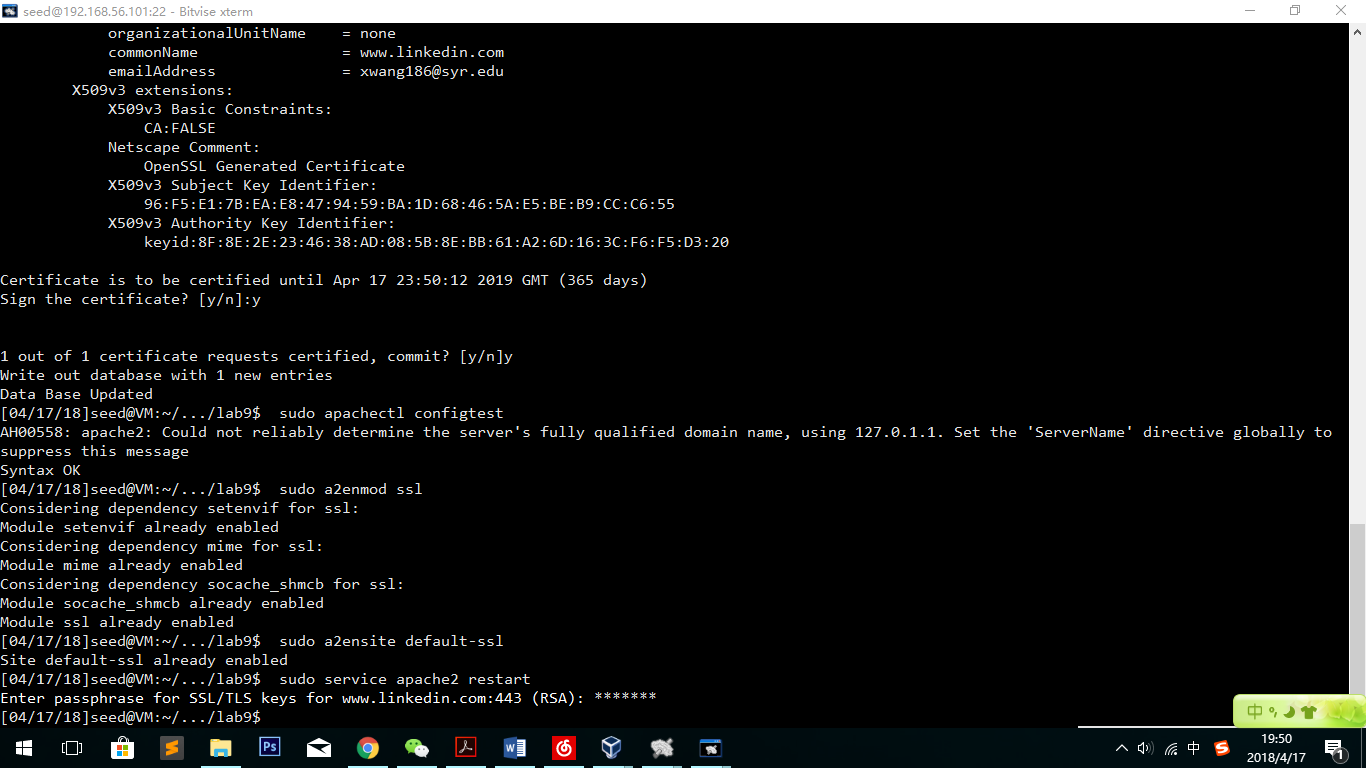
#### Test, reload and restart:

sudo apachectl configtest

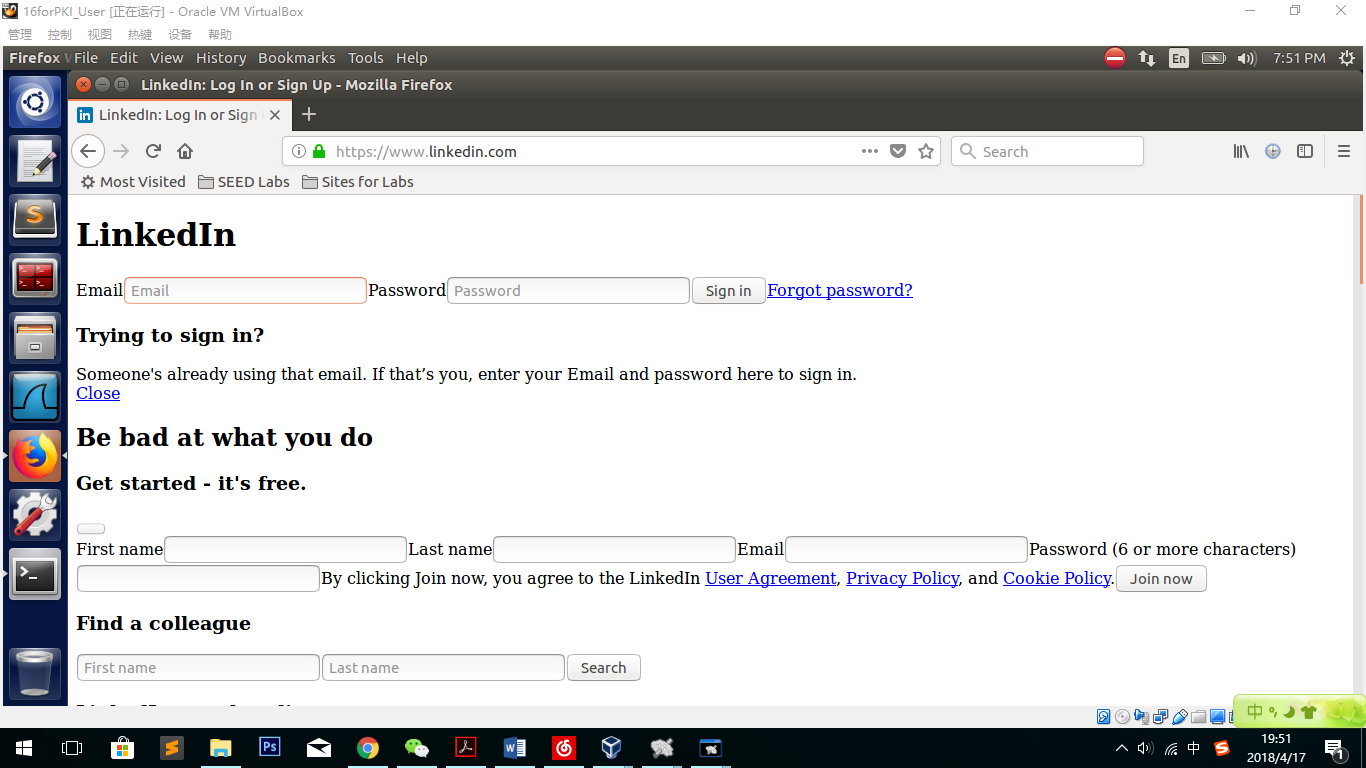
sudo a2enmod ssl

sudo a2ensite default-ssl

sudo service apache2 restart



#### Check the website again



**Observation:**

This time the https Man-In-The-Middle attack is successful. In the user’s machine, if the user has imported the ca file of the root CA and get DNS poisoned to redirect to 10.0.2.10 (which is the fake page’s ip address), the https attack will succeed no matter what the website name is.

**Explanation:**

This time, having the new signature in the fake page certified by the root CA’s private key, we can easily pass the browser’s certification. The root CA’s file has already been in the browser. So, we the https request appears, the browser checks the domain name and the signature, and find [www.linkedin.com](http://www.linkedin.com) they visit suits all the certification contents. That is why the browser thinks the fake website is secure.