

Please use a Screen or Video Capture software to save your works!

OBJECTIVE & PREPARATION

Prerequisites

This lab depends on changes made in several previous labs. You must have successfully completed labs on DNS, SMTP, LDA, LAMP and Roundcube in order to be able to do this lab.

Below is the same diagram that we referred to over the previous email labs:

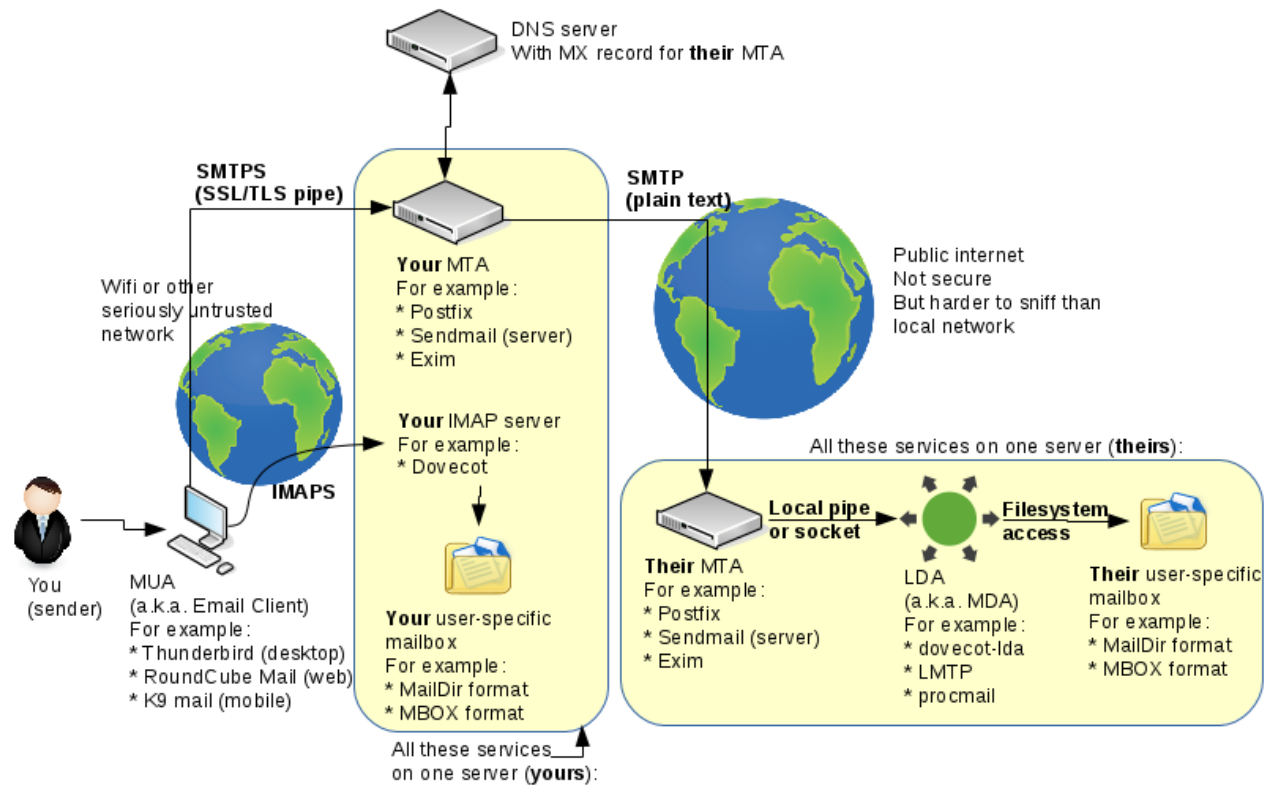


Diagram of **you** sending email to **them**

Note the two globes in the above diagram. Those globes represent the Internet that your emails travel through in order to be received by an e-mail recipient. **The smaller globe (the one your workstation is connected to) cannot be trusted to send mail messages unencrypted.** The **larger globe usually involves inter-ISP traffic, often through an internet trunk line, so it is also unencrypted, but it cannot be easily accessed by hackers, pen-testers, or evildoers.**

There are **two important general truths you need to understand about email encryption:**

- **Email (the way the majority of people use it) travels from SMTP server to SMTP server unencrypted.**
That means that nothing sent over email is truly secure. But attempting to continually intercept SMTP server to SMTP server traffic is difficult and expensive, not worth doing for the little bit of money most of us have in our bank account.
- **Email travelling over a LAN (especially Wifi, but any local network) is always encrypted.**
If e-mail traffic on a LAN was not encrypted, it would be easy and inexpensive to intercept (in order to obtain your username and password). These days, unencrypted connections from your client to your SMTP/IMAP/POP3 server very rarely exist.

You see in our diagram that one of the SMTP connections is supposed to be encrypted (this is the one that would be "LAN" traffic) and the IMAP connection as well (this one is either LAN-like traffic or is connecting to localhost, which is a different scenario altogether).

We're going to secure the two connections that we left to be in plain text previously. Unfortunately encrypting things is rarely a straightforward process.

Online Resources

[TSL, SSL Definition](#)

[Create a self signed SSL key for Postfix](#)

[Dovecot SSL configuration](#)

CASE 1: Generating a Self-Signed Certificate

Transport Layer Security (TLS) and its predecessor, **Secure Sockets Layer** (SSL), both of which are frequently referred to as 'SSL', are cryptographic protocols designed to provide communications security over a computer network.

Normally (in production), you would need to pay a "Certificate Authority" to issue a **certificate** for you. That is essentially a **"signed" public key** that will tell strangers on the internet that your server is really yours (i.e. the certificate authority says so). There is an obvious problem with the previous statement but that is mainly how public key encryption works on the Internet today.

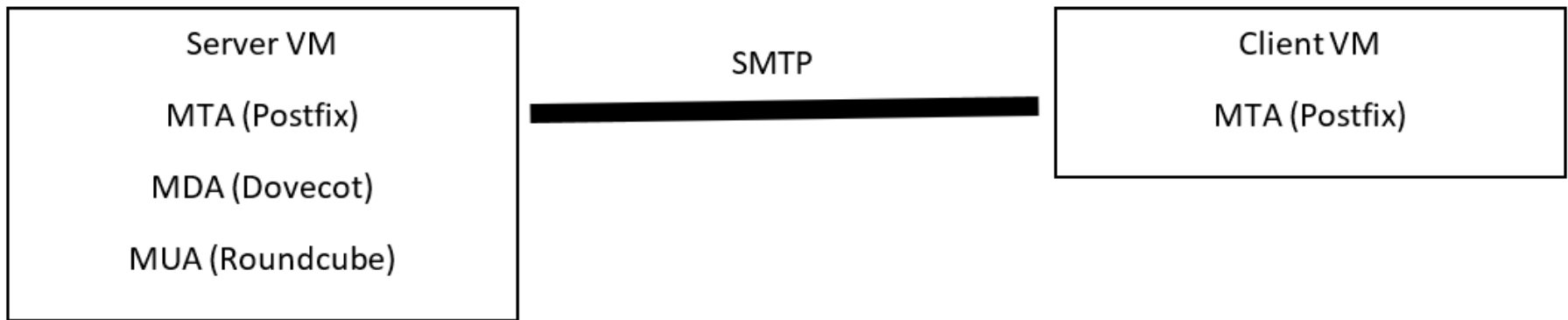
We will be generating our own public keys, mainly in order to avoid paying for a certificate. We will not have enough time to get into the details of what all the following commands do in this section. They are publicly accessible from Internet.

The public key cryptography concepts in this lab are the same in a previous lab (Lab1: SSH), although the terminology is slightly different.

A simple way to summarize the differences is:

- The **.key** file is your private key.
- The **.crt** file is your public key.

Encrypting Postfix with Transport Layer Security (TLS)



Perform the following steps:

1. Let's start with the "sending" SMTP server we have on **Client VM**. Run the following, replacing **jasonpang.ops** with your own domain name: (if asked for password, use our default **Password1234**)

```
mkdir -p /root/postfix-keys /etc/ssl/{private,certs}
```

```
cd /root/postfix-keys
```

```
openssl genrsa -des3 -out client.jasonpang.ops.key 2048
```

```
chmod 600 client.jasonpang.ops.key
```

```
openssl req -new -key client.jasonpang.ops.key -out client.jasonpang.ops.csr
```

```
openssl x509 -req -days 365 -in client.jasonpang.ops.csr -signkey client.jasonpang.ops.key -out client.jasonpang.ops.crt
```

```
openssl rsa -in client.jasonpang.ops.key -out client.jasonpang.ops.key.nopass
```

```
mv client.jasonpang.ops.key.nopass client.jasonpang.ops.key
```

```
openssl req -new -x509 -extensions v3_ca -keyout cakey.pem -out cacert.pem -days 365
```

```
chmod 600 client.jasonpang.ops.key cakey.pem
```

```
cp client.jasonpang.ops.key cakey.pem /etc/ssl/private
```

```
cp client.jasonpang.ops.crt cacert.pem /etc/ssl/certs
```

NOTE: Those commands will **create a certificate, a certificate signing request, a certificate authority, and sign your certificate with your certificate authority.**

This would be the same as in the real world except there you would contact a real CA, setup account and pay the fees. However, here you're making up your own CA, and all the services in your environment trust it.

2. Now, configure Postfix to use the generated certificate, by adding the following to your **main.cf** file:

Settings to enable secure SMTP using my self-signed certificate:

smtpd_tls_auth_only = no

smtpd_use_tls = yes

smtp_use_tls = yes

smtpd_tls_key_file = /etc/ssl/private/client.jasonpang.ops.key

smtpd_tls_cert_file = /etc/ssl/certs/client.jasonpang.ops.crt

smtpd_tls_CAfile = /etc/ssl/certs/cacert.pem

tls_random_source = dev:/dev/urandom

smtpd_tls_loglevel = 1

Setting Up and Testing Encryption with Thunderbird

Perform the following steps:

1. Currently your Thunderbird is set up to use client.yoursenecaid.ops for an SMTP server, with no security. Change that to use STARTTLS instead (you can change it under account settings --> Outgoing Server).
2. We haven't set up any user authentication, just an encrypted channel; therefore, leave the authentication method at the value: none.
3. When you try to send an email Thunderbird will warn you about the self-signed certificate. You obviously know it's your certificate so you can tell Thunderbird to trust it:

Your full name
Jason Pang

Email address
ops335@jasonpang.ops

Password
P@\$Sw0rd1234

☒ Remember password

✓ The following settings were found by probing the given server:

Manual configuration

INCOMING SERVER

Protocol: IMAP

Hostname: jasonpang.ops

Port: 143

Connection security: None

Authentication method: Normal password

Username: ops335

OUTGOING SERVER

Hostname: client.jasonpang.ops

Port: 25

Connection security: STARTTLS

Authentication method: No authentication

Username: ops335

Advanced config

Re-test Cancel Done

Send Message Error

✖

! Sending of the message failed.
The certificate is not trusted because it is self-signed.
The configuration related to client.jasonpang.ops must be corrected.

OK

Add Security Exception

✖

! You are about to override how Thunderbird identifies this site.
Legitimate banks, stores, and other public sites will not ask you to do this.

Location: client.jasonpang.ops:25 [Get Certificate](#)

This site attempts to identify itself with invalid information. [View...](#)

Wrong Site
The certificate belongs to a different site, which could mean that someone is trying to impersonate this site.

Unknown Identity
The certificate is not trusted because it hasn't been verified as issued by a trusted authority using a secure signature.

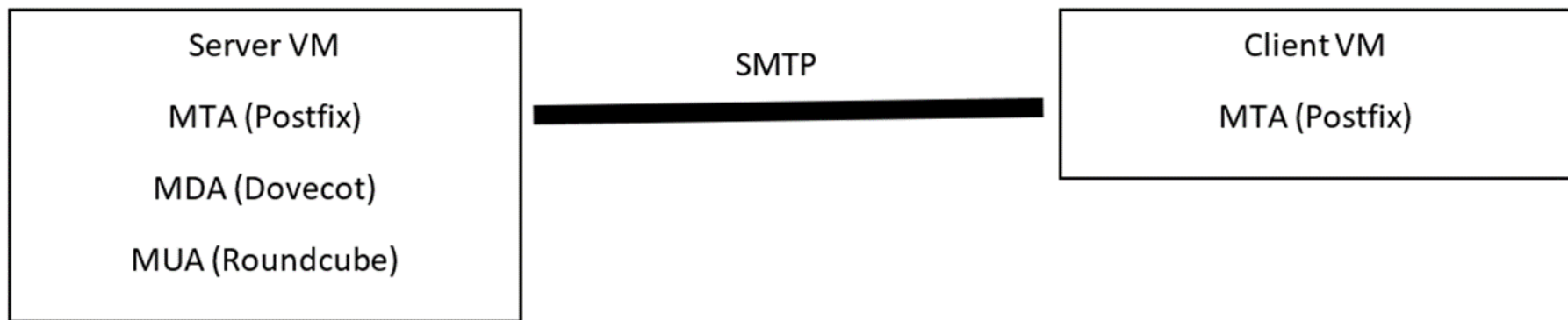
☒ **Permanently store this exception**

[Confirm Security Exception](#) Cancel

NOTE: Your message may look slightly different.

4. After you confirm that security exception, send another email to yourself and make sure you receive it.
5. Notice that from the user's point of view nothing is different. But if you were an evildoer trying to steal an identity (the difference is huge). Before it was trivial and now it's computationally prohibitive.

Encryption Dovecot with Secure Socket layer (SSL)



Now we will ensure that our **Dovecot** connection is secure, and enforce that policy. With SMTP, you will need to allow plain text connections since that is the only method to pass email from server-to-server. With IMAP, there is no server-to-server interaction, but rather only client-to-server interaction. The reason to have an unencrypted IMAP connection would be if your **IMAP server** and **IMAP client** were the same machine.

Perform the following steps:

1. Let's start by generating a new certificate for Dovecot on your **server** machine by issuing the following commands: (if asked for password, use our default **Password1234**)

```
mkdir -p /root/postfix-keys /etc/ssl/{private,certs}
```

```
cd /root/postfix-keys
```

```
openssl genrsa -des3 -out server.jasonpang.ops.key 2048
```

```
chmod 600 server.jasonpang.ops.key
```

```
openssl req -new -key server.jasonpang.ops.key -out server.jasonpang.ops.csr
```

```
openssl x509 -req -days 365 -in server.jasonpang.ops.csr -signkey server.jasonpang.ops.key -out server.jasonpang.ops.crt
```

```
openssl rsa -in server.jasonpang.ops.key -out server.jasonpang.ops.key.nopass
```

```
mv server.jasonpang.ops.key.nopass server.jasonpang.ops.key
```

```
openssl req -new -x509 -extensions v3_ca -keyout cakey.pem -out cacert.pem -days 3650
```

```
chmod 600 server.jasonpang.ops.key cakey.pem
```

```
cp server.jasonpang.ops.key cakey.pem /etc/ssl/private
```

```
cp server.jasonpang.ops.crt cacert.pem /etc/ssl/certs
```

NOTE: This process is **identical** to what you've done for the **Client VM** certificate. In fact, if your IMAP and SMTP servers are on the same machine you can share the certificate between them. In our case, they are not on the same machine.

2. Next, we need to configure Dovecot to use this for encrypted connections and not allow any kind of plain text connections. Edit the **10-auth.conf**, and **10-ssl.conf** files and change the following settings (note: these parameters already exist in those files, just find them and set them to the correct value):

[/etc/dovecot/conf.d/10-auth.conf](#)

`disable_plaintext_auth = yes`

[/etc/dovecot/conf.d/10-ssl.conf](#)

`ssl = required`

`ssl_cert = </etc/ssl/certs/server.jasonpang.ops.crt`

`ssl_key = </etc/ssl/private/server.jasonpang.ops.key`

3. Now, we will disable normal IMAP connections, leaving only IMAPS (secured IMAP) allowed. Edit the 10-master.conf file and set the port number in inet_listener imap to 0. And enable imaps.

[/etc/dovecot/conf.d/10-master.conf](#)

```
service imap-login {  
  inet_listener imap {  
    port = 0  
  }  
  inet_listener imaps {  
    port = 993  
    ssl = yes  
  }  
}
```

4. Your key/certificate doesn't have a **“.pem”** extension but they are PEM-encoded files. You can confirm that using the **file** command. If you're interested, learning more about configuring Dovecot for SSL, refer to the following documentation: [Dovecot SSL configuration](#).

`file /etc/ssl/certs/server.jasonpang.ops.crt`

`file /etc/ssl/private/server.jasonpang.ops.key`

```
[root@server postfix-keys]# file /etc/ssl/certs/server.jasonpang.ops.crt
/etc/ssl/certs/server.jasonpang.ops.crt: PEM certificate
[root@server postfix-keys]# file /etc/ssl/private/server.jasonpang.ops.key
/etc/ssl/private/server.jasonpang.ops.key: PEM RSA private key
```

5. You better restart the following services before move to next step.

`systemctl restart postfix`

`systemctl restart httpd`

`systemctl restart dovecot`

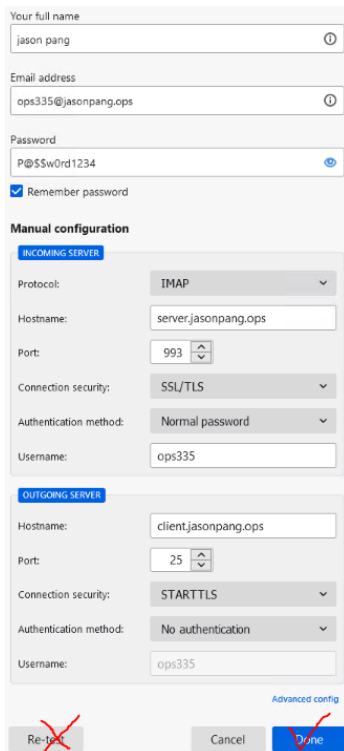
Verifying that Mail Messages are Encrypted

Perform the following steps:

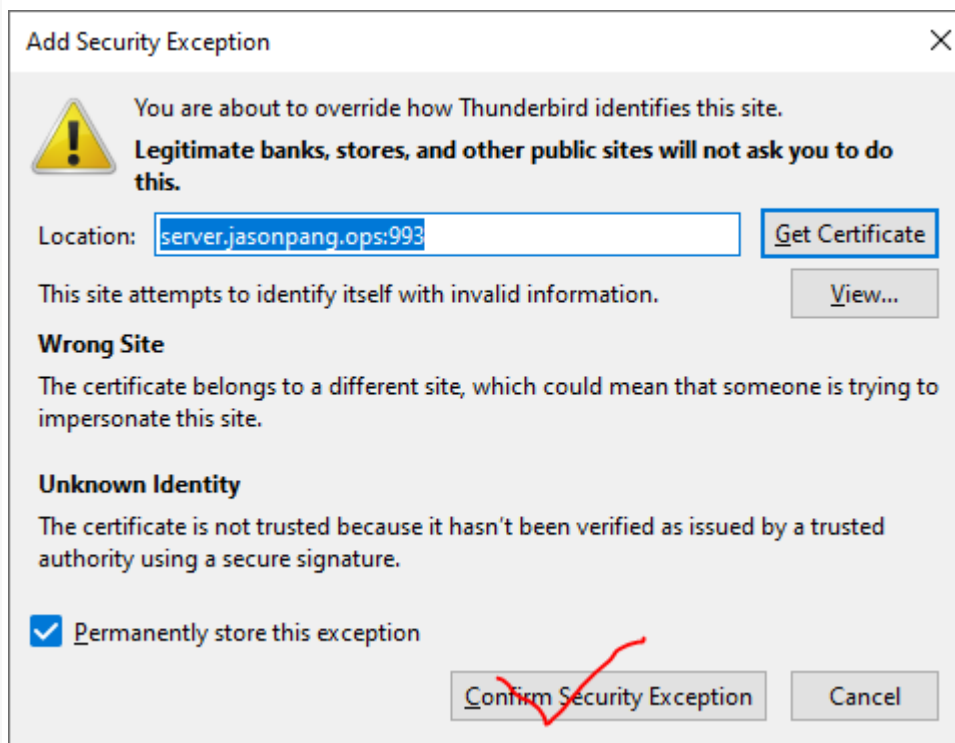
1. Use the **ss -apnt** command to confirm you're only listening on the IMAPS port 993, and not the plaintext IMAP port 143.
2. Next, reconfigure your account settings in Thunderbird to use the **SSL/TLS** connection security with your IMAP server, leaving the password as **Normal Password**.

NOTE: When you send your test email, you will get another warning because you're using a self-signed certificate on client.**jasonpang**.ops . Make sure to authorize the exception.

You don't need to re-test the settings, it won't work because you are using a self-signed certificate.



Thunderbird account configuration window showing settings for 'jason pang' (ops335@jasonpang.ops). The account is configured for IMAP (server.jasonpang.ops, port 993) and STARTTLS (client.jasonpang.ops, port 25). The connection security is set to SSL/TLS for IMAP and STARTTLS for STARTTLS. The authentication method is set to Normal password for IMAP and No authentication for STARTTLS. The username is ops335 for both. The 'Remember password' checkbox is checked. The 'Advanced config' link is visible at the bottom right.



Add Security Exception dialog box. The location is server.jasonpang.ops:993. The dialog warns that the site attempts to identify itself with invalid information and that the certificate belongs to a different site. The 'Get Certificate' button is highlighted. The 'View...' button is also visible. The 'Wrong Site' and 'Unknown Identity' sections provide further details. The 'Permanently store this exception' checkbox is checked. The 'Confirm Security Exception' button is highlighted with a red checkmark, and the 'Cancel' button is also visible.

Case 2: Configure Webmail (Roundcube Mail) To Use Encryption

In this case, we will modify the **roundcube** webmail application to make use of the encrypted connections the email servers provide, and to allow clients to connect to it using an encrypted connection (**https**).

Perform the following steps on Server VM:

First, we will modify the webserver to use the encrypted connections the email servers are now providing

1. Modify the roundcube configuration file </var/www/html/webmail/config/config.inc.php> so that the following parameters reflect the encrypted settings on the mail servers:

```
$config['smtp_server'] = 'client.jasonpang.ops';
```

```
$config['default_host'] = 'ssl://server.jasonpang.ops';
```

```
$config['default_port'] = 993;
```

NOTE: The last two entries above refer to your IMAPS server

2. You may wish to use the configuration tool in your Roundcube installer after making those changes. To do so, you will need to add the following to your configuration file.

```
$config['enable_installer'] = true;
```

Next, test if the roundcube webmail application is working by sending and receiving e-mail messages.

(make sure you restart the **httpd** service)

Now that the webmail application is using an encrypted connection when communicating with the email servers, it is time to encrypt the client's connection to the web server.

1. First you need to generate a new certificate for apache on your Server VM. (You have the Web server on the Server VM, which has already created a certificate from the above steps, you can simply use it. If you are hosting the webserver on a different VM, you have to generate a new one with the same steps from above.) Certificate is Binding to the FQDN hostname. i.e. server.jasonpang.ops & client.jasonpang.ops
2. Install the **mod_ssl** package to allow apache to use ssl.
3. Add the following parameters to the apache configuration file: [vi /etc/httpd/conf/httpd.conf](#)

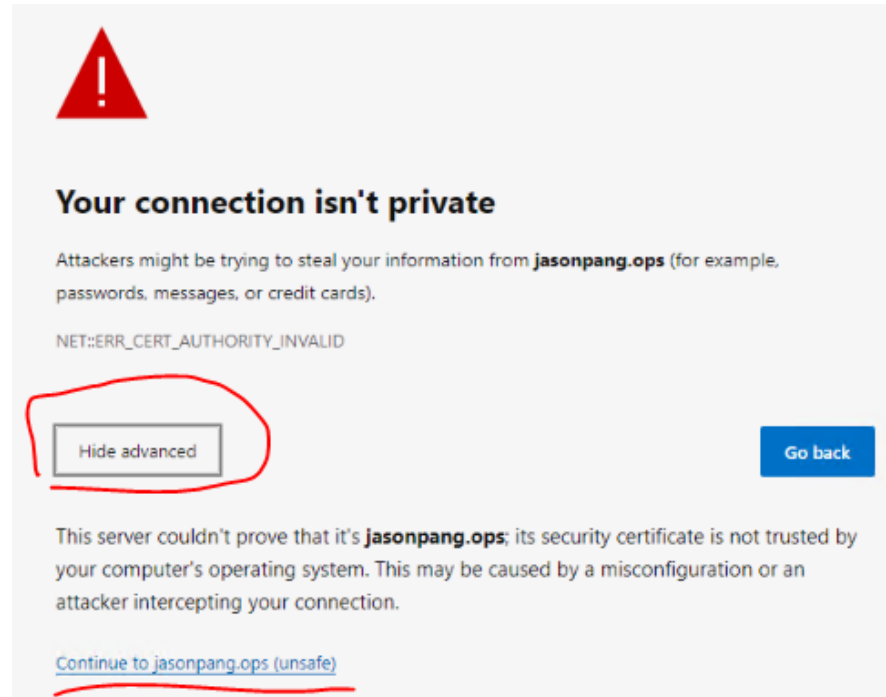
SSLEngine on

SSLCertificateFile "/etc/ssl/certs/server.jasonpang.ops.crt"

SSLCertificateKeyFile "/etc/ssl/private/server.jasonpang.ops.key"

4. Restart apache and modify your firewall to allow traffic to **port 443**.
5. Open a web-browser on your host and try to connect to [https://\[server.\]<yourdomain>.ops/webmail](https://[server.]<yourdomain>.ops/webmail)
[server.] is optional if you have your DNS set properly.

You should get a security exception similar to the one's you saw with the email, and for the same reason (the site you are trying to contact has a self-signed certificate). Add the exception and login to access your email. Send an email to ensure everything is functioning properly.



Once everything tested working. Remove the newly created Windows VM.

Save the captured file(s) as OPS345_Lab07_yourusername and upload to Blackboard.

If it is video recordings, upload to OneDrive and share with jason.pang@senecacollege.ca