LetsEncrypt and ISPConfig

As of ISPConfig v3.3, there is built in support for LetsEncrypt using the acme.sh script. The following is old information for ancient versions of ISPConfig, before they did this.

If you are using any modern version of ISPConfig, **do not do the following**. I am leaving it here for a while, just in case someone needs it.

DO NOT USE THIS

ISPConfig (http://ispconfig.org) is a very nice control panel for Linux. It does not work well in other Unicies, but is almost seamless with Debian and Devuan. ISPConfig works well with several servers (mail,web, etc...), which you choose at installation time. Our setup uses Apache2, Postfix and Dovecot, which this article is written for.

The installation script will set up your server(s) and, if you agree, set up self-signed certs for your web/smtp/imap/pop servers. This article discusses replacing those certs with certs provided by LetsEncrypt (https://letsencrypt.org).

NOTE: There is an updated script from the ISPConfig people themselves which gives a nice walk through. I'd recommend reading it and, probably, use it instead of this document. Check it out at https://www.howtoforge.com/tutorial/securing-ispconfig-3-with-a-free-lets-encrypt-ssl-certificate/

Setting up the Apache web server

certbot is a pretty decent little installer, and it knows Apache2. It really helps to use the automated tool until you get a chance to figure out all the ins and outs of LetsEncrypt. However, the installer is a little touchy on Debian systems when it tries to install some packages; if your APT sources have errors, you can have more troubles than you need, so you should verify your system beforehand.

Verify your system

I strongly recommend you set up backports before using the installer, and make sure it is all working well. Run the following two commands, and watch *apt-get update* closely for any errors. Fix any errors (by removing repositories or fixing them) before proceeding.

```
echo "deb http://ftp.debian.org/debian wheezy-backports main" >
/etc/apt/sources.list.d/backports.list
apt-get update
```

Install certbot and run it

I tend to put optional software in /opt, so we'll create a directory named certbot in /opt, download the

installer, then run it.

When you get to the point where it asks which virtual to use, select the virtual with the same name as your actual server.

Also, I chose “simple” on the install type. That allows http and https.

install_certbot.sh

```
mkdir -p /opt/certbot
cd /opt/certbot/
wget https://dl.eff.org/certbot-auto
chmod a+x certbot-auto
./certbot-auto
```

If you get an error, check out this article for one solution.

ISPConfig specialized configuration

When you have done the above, certbot will have created a new container for you in /etc/apache2/sites-available. An example of the new vhost container name would be *mail.example.com.vhost-le-ssl.conf*. Look in that file (/etc/apache2/sites-available/mail.example.com.vhost-le-ssl.conf). Near the bottom, you will see the following lines:

```
SSLCertificateFile /etc/letsencrypt/live/mail.example.com/cert.pem
SSLCertificateKeyFile /etc/letsencrypt/live/mail.example.com/privkey.pem
Include /etc/letsencrypt/options-ssl-apache.conf
SSLCertificateChainFile /etc/letsencrypt/live/mail.example.com/chain.pem
```

These are the lines you should include in your ISPConfig vhost file for the administrative interface. That file is ispconfig.vhost. Open that file (/etc/apache2/sites-available/ispconfig.vhost) and search for the string *SSL Configuration*. On our machine it looks like:

```
  # SSL Configuration
  SSLEngine On
  SSLProtocol All -SSLv2 -SSLv3
  SSLCertificateFile /usr/local/ispconfig/interface/ssl/ispserver.crt
  SSLCertificateKeyFile /usr/local/ispconfig/interface/ssl/ispserver.key
  #SSLCACertificateFile /usr/local/ispconfig/interface/ssl/ispserver.bundle
```

Comment out the two certificate file names, and add the information from the certbot install:

```
  # SSL Configuration
  SSLEngine On
  SSLProtocol All -SSLv2 -SSLv3

  # letsencrypt certbot files 20160925 by me
  SSLCertificateFile /etc/letsencrypt/live/mail.example.com/cert.pem
```

```
  SSLCertificateKeyFile /etc/letsencrypt/live/mail.example.com/privkey.pem
  Include /etc/letsencrypt/options-ssl-apache.conf
  SSLCertificateChainFile /etc/letsencrypt/live/mail.example.com/chain.pem
# end of letsencrypt certbot files

  #SSLCertificateFile /usr/local/ispconfig/interface/ssl/ispserver.crt
  #SSLCertificateKeyFile /usr/local/ispconfig/interface/ssl/ispserver.key
  #SSLCACertificateFile /usr/local/ispconfig/interface/ssl/ispserver.bundle
```

certbot is smart enough to know about Debian Apache. Configuration files are created in /etc/apache2/sites-available, then the ones you want to be run are linked (symbolic link) to /etc/apache2/sites-enabled. When certbot created the mail.example.com.vhost-le-ssl.conf vhost file, it linked it to sites-enabled. So, simply remove it.

```
rm /etc/apache2/sites-enabled/mail.example.com.vhost-le-ssl.conf
/etc/init.d/apache2 restart # or, service apache2 restart
```

You should now be able to access your control panel at http://mail.example.com:8080 with no certificate errors.

Setting your mail to use the Certs

Setting up the mail servers is very dependent on how the mail servers were configured. Since this article is on ISPConfig, we'll take the default for them, but the same applies to other mail servers. If you want a quick and dirty, simply use the script below.

Generalized Script

The following script works on my installation of ISPConfig. You can simply download this and use it if you are sure the postfix and dovecot certs are in the same place mine are.

Be sure to change *server.example.com* to be the actual server name your installation used (hint, look in /etc/letsencrypt/live/).

mailcerts.sh

```
#! /bin/bash

SERVERNAME=server.example.com

# postfix first
mv /etc/postfix/smtpd.cert /etc/postfix/smtpd.cert.save
mv /etc/postfix/smtpd.key /etc/postfix/smtpd.key.save
ln -s /etc/letsencrypt/live/$SERVERNAME/privkey.pem
/etc/postfix/smtpd.key
ln -s /etc/letsencrypt/live/$SERVERNAME/fullchain.pem
/etc/postfix/smtpd.cert
/etc/init.d/postfix restart
```

```
# now, dovecot
mv /etc/dovecot/dovecot.pem /etc/dovecot/dovecot.pem.save
mv /etc/dovecot/private/dovecot.pem
/etc/dovecot/private/dovecot.pem.save
ln -s /etc/letsencrypt/live/$SERVERNAME/fullchain.pem
/etc/dovecot/dovecot.pem
ln -s /etc/letsencrypt/live/$SERVERNAME/privkey.pem
/etc/dovecot/private/dovecot.pem
/etc/init.d/dovecot restart
```

How it works

Postfix and Dovecot have the ability to store the certificates in user defined locations. letsencrypt's client created two files:

- privkey.pem - the key file
- fullchain.pem - the certificate file

These need to be linked to the appropriate files for the server you want to use.

Postfix

If you want to locate the certs for Postfix, look in main.cf, or run the following command:

```
grep 'smtpd_tls_.*file' /etc/postfix/main.cf | grep -v '^#'
```

On our system, this returns

```
smtpd_tls_cert_file = /etc/postfix/smtpd.cert
smtpd_tls_key_file = /etc/postfix/smtpd.key
```

Which are the files that need to be replaced. I rename them with a .bak suffix, then simply create a symbolic link to the letsencrypt installed.

Dovecot

For Dovecot, it is the same, though ISPConfig uses the same file name for the key and the cert, but puts the key in the /etc/dovecot/private directory for protection. However.

```
egrep -r 'ssl_key|ssl_cert' /etc/dovecot/conf.d/10-ssl.conf | grep -v '^#'
```

Again, on our machine it returns

```
ssl_cert = </etc/dovecot/dovecot.pem
ssl_key = </etc/dovecot/private/dovecot.pem
```

Which are the files which need to be moved, then created as symbolic links.

Citations

- https://community.letsencrypt.org/tutorial-on-ejabberd-postfix-dovecot-and-or-nginx-with-letsencrypt/7320
- https://backports.debian.org/Instructions/