OLUG Presentation

Let's Encrypt at Home

March 5, 2024

By Aaron Grothe

Introduction

Let’s Encrypt at Home?

You can create self-signed certificates pretty easily for home. We’re going to do that in a couple of minutes.

The hard part is having a signature on the certificate that is recognized by web browsers.

There are only a handful of Certificate Authorities that are recognized by the popular web browsers.

Introduction (Continued)

If you have questions/comments please feel free to ask them anytime. You don't have to hold them until the end of the talk.

If you have anything to add to the talk, please feel free to speak up.

I've only been doing the signed certs in the lab for about a month or so. I've been using LetsEncrypt on NEbraskaCERT for something like 8+ years.

Hopefully this will be an interactive and productive session.

How Let’s Encrypt Worked CIRCA 2015

The one thing that Let’s Encrypt really needs is for you to verify that you own/control the domain.

E.g. I can’t be able to go and claim an SSL certificate for bigriver.com and then start selling books using that certificate. That would be bad

How Let's Encrypt Worked CIRCA 2015

So how it used to work

- You'd have a folder on the website where the Let's Encrypt (LE) Certificate Authority (CA) would put its confirmation file
- The certbot program would tell the CA to retrieve the special file
- This would denote that you controlled the Server and a certificate could be signed by LE with pretty good confidence

This is how it is working on NEbraskaCERT's https://www.nebraskacert.org webserver to this day :-)

Why that's suboptimal for a Home Lab

1. You probably aren't going to have a routable IP address from the internet

E.g. your homelab machines typically have a non-routable IP address: typically 192.168.x.x or 10.x.x.x. These aren't accessible from the internet.

You can use the IP address you get from your ISP. This might not be consistent and then you get to play with Dynamic DNS. Small plug for noip.com here

Why that's suboptimal for a Home Lab

2. You'd have to have a folder on your home network accessible from the internet

E.g. Most routers have the ability to mount a USB drive and share it to the internet. Or you can forward your traffic from the router to a machine in your home network.

You don't want to put your home network more on the internet than needed.

Why that’s suboptimal for a Home Lab

3. Your ISP may block port 80 and 443 access to your home network.

E.g. Many ISPs are blocking access to ports 80 and 443 to their residential customers. The cynical part of me might say this is part of their scheme to get people to buy their “business” plans

You may not have this an option

So What is the Alternative?

The alternative is you can have the certbot program update your dns record with a TXT record that confirms you control the domain.

- You don't have to have an ip address reachable from the internet
- You don't have to share anything from your home network to the internet
- You don't have to worry about what your ISP blocks inbound
- This is handled pretty easily by the certbot program

Some of the Supported DNS plugins

- certbot-dns-cloudflare
- certbot-dns-dnsimple
- certbot-dns-gehirn
- certbot-dns-linode
- certbot-dns-nsone
- certbot-dns-rfc2136
- certbot-dns-sakuracloud

certbot-dns-digitalocean
certbot-dns-dnsmadeeasy
certbot-dns-google
certbot-dns-luadns
certbot-dns-ovh
certbot-dns-route53

And there are many more for sites such as Hurricane Electric and so on

https://github.com/search?q=certbot%20dns&type=repositories

DEMO!!!

Demo will be in the following steps

1. Enable https on the test webserver with a self-signed certificate. Works, but is not what we want
2. Register our domain with our DNS server (duckdns.org)
3. Install certbot on our system
4. Generate a signed cert for our testbox
5. Install the signed cert
6. Test
7. Automate renewals

Enable SSL on our Webserver

We've got the default http webserver installed on our debian box. Time to enable SSL

```
### turn on apache ssl

% a2enmod ssl
% a2ensite default-ssl
% service apache2 reload
```

Enable SSL on our Webserver

We've got the default http webserver installed on our debian box. Time to enable SSL

```
### generate an SSL certificate

% openssl req -x509 -nodes -days 3650 -newkey rsa:2048 \
        -keyout /etc/apache2/ssl/olugdemo.key \
        -out /etc/apache2/ssl/olugdemo.crt
```

And deal with the prompts

Enable SSL on our Webserver

We've got the default http webserver installed on our debian box. Time to enable SSL

```
### install SSL certificate into webserver

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

	SSLCertificateFile	/etc/apache2/ssl/olugdemo.crt
	SSLCertificateKeyFile
/etc/apache2/ssl/olugdemo.key

service apache2 reload
```

Enable SSL on our Webserver.

Time to hit the website and verify that is has SSL :-)

Warning that the certificate isn't trusted.

So now it is time to install certbot and sign the certificate

Register our Domain with DuckDNS.org

We’ll go with DuckDNS.org for our domains

Why?

DuckDNS.org is quick, easy, free (for up to 5 addresses) and has a plugin for certbot :-)

We’ll start with needing the IP address of the system

```
# ip -4 a
```

Register our Domain with DuckDNS.org

So now we need to set the ip address for the domain name we're using

In this case it'll be the olugdemo.duckdns.org

So now it is time to test that it is working for us

HTTPS is working but still not working without the warning

Time to Fix that

Time to Setup Certbot

We need to setup a python environment to allow us to install Python on our box

There are two main ways to do it

#1. Direnv/Python

#2. Setup a virtual environment using the version of python that comes with the system

We'll go with #2 for this presentation

Time to Setup Certbot

Install python3-pip

```
# apt-get install python3-pip
# apt install python3.11-venv

% python3 -m venv /home/grothe/certbot

% PATH=/home/grothe/certbot/bin:$PATH
% export PATH
```

Also time to add this to .bashrc

Time to Setup Certbot

Install certbot and certbot_dns_duckdns

```
$ pip install certbot
$ pip install certbot_dns_duckdns
```

Time to check and make sure that certbot works

```
$ certbot –version
```

Check installed versions

```
$ pip list | grep certbot
```

Time to Setup Certbot

Time to generate a LE cert

```
sudo /home/grothe/python/bin/certbot certonly \
  --non-interactive \
  --agree-tos \
  --email ajgrothe@yahoo.com \
  --preferred-challenges dns \
  --authenticator dns-duckdns \
  --dns-duckdns-token
c075b8d0-7e24-45ab-8a6b-795dd362cc00 \
  --dns-duckdns-propagation-seconds 120 \
  -d "*.olugdemo.duckdns.org"
```

Time to Setup Certbot

Let's go through the command

```
# Run the command as root

sudo /home/grothe/python/bin/certbot certonly \

# don't ask me for any input

  --non-interactive \
```

Time to Setup Certbot

Continuing

```
# Agree to the EULA
  --agree-tos \
# My email address, requested by LE
  --email ajgrothe@yahoo.com \
```

Time to Setup Certbot

Continuing

```
# DNS challenge
  --preferred-challenges dns \
# Use the DuckDNS certbot plugin
  --authenticator dns-duckdns \
```

Time to Setup Certbot

Contiuining

```
# Duckdns token

  --dns-duckdns-token
c075b8d0-7e24-45ab-8a6b-795dd362cc00 \

# How long to wait for the DNS record to update

  --dns-duckdns-propagation-seconds 120 \
```

Time to Setup Certbot

Concluding

# We're generating a wildcard Certificate for the system

So this will cover sites such as plex.olugdemo.duckdns.org, test.olugdemo.duckdns.org and so on

```
  -d "*.olugdemo.duckdns.org"
```

Time to Install the Certificate

Verify contents of the directory

```
# ls /etc/letsencrypt/live/olugdemo.duckdns.org

cert.pem
chain.pem
fullchain.pem
privkey.pem
README
```

Installing the Cert

```
### install SSL certificate into webserver

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

        SSLCertificateFile
/etc/letsencrypt/live/olugdemo.duckdns.org/fullchain.pem
        SSLCertificateKeyFile
/etc/letsencrypt/live/olugdemo.duckdns.org/privkey.pem

service apache2 reload
```

Moment of Truth

Open a web browser and go to

https://olugdemo.duckdns.org

Success - bow and wait for applause
Failure - figure out how to blame it on Microsoft

So What have We Done So Far

- Enabled SSL on the webserver
- Generated a self-signed certificate
- Put an entry into Duckdns.org for our homelab
- Installed Python into a venv on our Debian box
- Generated an LE certificate for our domain
- Installed the LE certificate on our webserver

What’s Next

Let’s Encrypt certificates are only good for 90 days. This is done for a security measure and also to encourage people to exercise good automation processes

Running “certbot renew” will renew certs when they get near their expiration time

Let’s go ahead and create a small script to handle the renewal for us

Renewal Script

```
% cat /etc/cron.daily/certbot
```

```
#!/bin/sh

/home/grothe/python/bin/certbot renew > /dev/null 2>&1
```

And make the script live

```
% chmod +x /etc/cron.daily/certbot
```

That should be it for us

Alternatives / Tools

There are a lot of alternatives we didn't discuss for the parts

Webserver - We used apache, you can use nginx, lighttpd or anything else. As long as it does certificates in a similar matter you'll be able to put your certs into the system

DuckDNS - There are quite a few free DNS systems out there, you can try a lot of them, or if you want to run your own DNS system. I went with DuckDNS because it was the easiest

Alternatives / Tools

There are a lot of alternatives we didn't discuss for the parts

Certbot - We used the certbot program directly because it is the system I've had the most luck with. There are a ton of other programs that will work with the ACME protocol that LE uses. There are versions written in go, in bash and some versions such as acme-tiny-dns which does the majority of the Certbot functionality in about 200 lines of code

Different Options

I did my example by hand as that is how I have it setup in my home lab.

There is also a very nice nginx/reverse proxy/docker container setup that makes it a lot easier to do a lot of this

It is called the Nginx Proxy Manager

We'll hit their webpage a bit

https://nginxproxymanager.com/

Let's Reset our Machine and do a Demo

Steps in Demo

- Reset the machine
- Stop the apache webserver on the box
- Install docker and docker-compose
- Create a docker compose file for Nginx Proxy Manager (NPM)
- Bring up the system
- Create a quick proxy pointer to the homepage for the NPM

Reset the Machine

Just going to restore our snapshot of the Demo machine

Can do either through the gui or by the command line

We'll do the GUI for this round

Confirm that SSL isn't on for the webserver

https://olugdemo.duckdns.org - it isn't working. Good.

Stop the Apache Server

We are running the non-secure webserver on port 80.

Nginix Proxy Manager is going to want port 80, so we need to stop and disable the apache2 webserver

```
# systemctl stop apache2
# systemctl disable apache2
```

Install Docker and Docker-compose

Need to install docker and docker-compose

```
# apt install docker.io
# apt install docker-compose
```

docker.io is how the package is defined in debian, because of a conflict with an old window manager

Add our user to the docker group in /etc/group, and logout/login

Create a basic docker-compose file

```yaml
version: '3.8'
services:
  app:
    image: 'jc21/nginx-proxy-manager:latest'
    restart: unless-stopped
    ports:
      - '80:80'
      - '81:81'
      - '443:443'
    volumes:
      - ./data:/data
      - ./letsencrypt:/etc/letsencrypt
```

Time to bring up the new environment

```
# docker-compose up -d
```

Wait a bit for it to finish downloading and starting everything

Connect to the system at port 81 and start configuring

https://www.olugdemo.duckdns.org:81

Login: admin@example.com
Password: changeme

Time to bring up the new environment

```
# docker-compose up -d
```

Wait a bit for it to finish downloading and starting everything

Connect to the system at port 81 and start configuring

https://www.olugdemo.duckdns.org:81

Login: admin@example.com
Password: changeme

Need to change the password

Change the passwords from defaults

Username: admin@example.com
Password: changeme

Generate a Let's Encrypt cert

We'll step the process of creating a Let's Encrypt certificate for the NPM

Keep in mind you may have to do this multiple times to get it to work. It timed out on my several times.

Note: dropdown of all the available plugins gives you an idea of the scope of options available.

We'll create a Proxy

Lets point npm.olugdemo.duckdns.org to the Nginx Proxy Manager

So let's create a quick proxy

Destination http://localhost:81
SSL - Let's Encrypt

Time to test it out

https://npm.olugtest.duckdns.org

Next Steps

So the major thing is we've added additional points. If you want to restrict access to services to https only there are a couple of options

1. Change the config files for the services. For example, you can change the listening port for Apache to only listen to requests from the proxy server
2. You can put in firewall rules to restrict access to the unsecure services. E.g. lock down NPM so it only allows access to port 81 from the localhost.

We'll leave this an exercise for the reader

Three Things I Wish I had Known

Three things I wish I had known before I started

1. DNS propagation timeout is by default 30 seconds, up it to 120 seconds and you will have better success
2. Nginx Proxy Manager - works well if you're controlling everything through the Docker, mixing docker and non-docker instances seems a bit dicey. Lot of 502 bad gateway errors
3. Make sure you grab the wildcard and regular domain in your request, otherwise you might be wondering why www.olugdemo.duckdns.org is working why olugdemo.duckdns.org isn't

Three Tips

Three tips for those giving this a spin

1. Lets Encrypt has limits on how many certificate requests you can do in a day. It is approximately 300 per 3 hours, so be careful when scripting. There is a staging environment you can use as well.
2. You can combine duckdns.org with your own local dns in case you want to make sure you have access without hitting the internet for DNS all the time
3. You can change your DuckDNS API by selecting the ||| token and changing the API

Summary

So that is a quick introduction to how you can get an SSL certificate setup for your home lab.

It has been a pretty quick intro, but think we gave a decent overview

Slides will be on my website https://www.grothe.us on my presentations page

Thanks for listening :-)

Links

Let's Encrypt Home Page

https://www.letsencrypt.org

Duck DNS

https://www.duckdns.org

Certbot

https://certbot.eff.org

Links

Certbot DuckDNS plugin

https://github.com/infinityofspace/certbot_dns_duckdns

ACME Tiny DNS - Alternative to Certbot

https://github.com/Trim/acme-dns-tiny/

Nginx Proxy Manager

https://nginxproxymanager.com/

Links

Couple of Youtube talks - using Nginx Proxy Manager, helped me figure out a few things

https://www.youtube.com/watch?v=qlcVx-k-02E&t=435s&pp=ygUVbGV0cyBlbmNyeXB0IGhvbWUgbGFi

https://www.youtube.com/watch?v=sRI4Xhyedw4&t=2s&pp=ygUVbGV0cyBlbmNyeXB0IGhvbWUgbGFi