Create Your Own SSL Certificate Authority for Local HTTPS Development

(https://deliciousbrains.com/ssl-certificate-authority-for-local-https-development/) PUBLISHED NOV 23, 2021

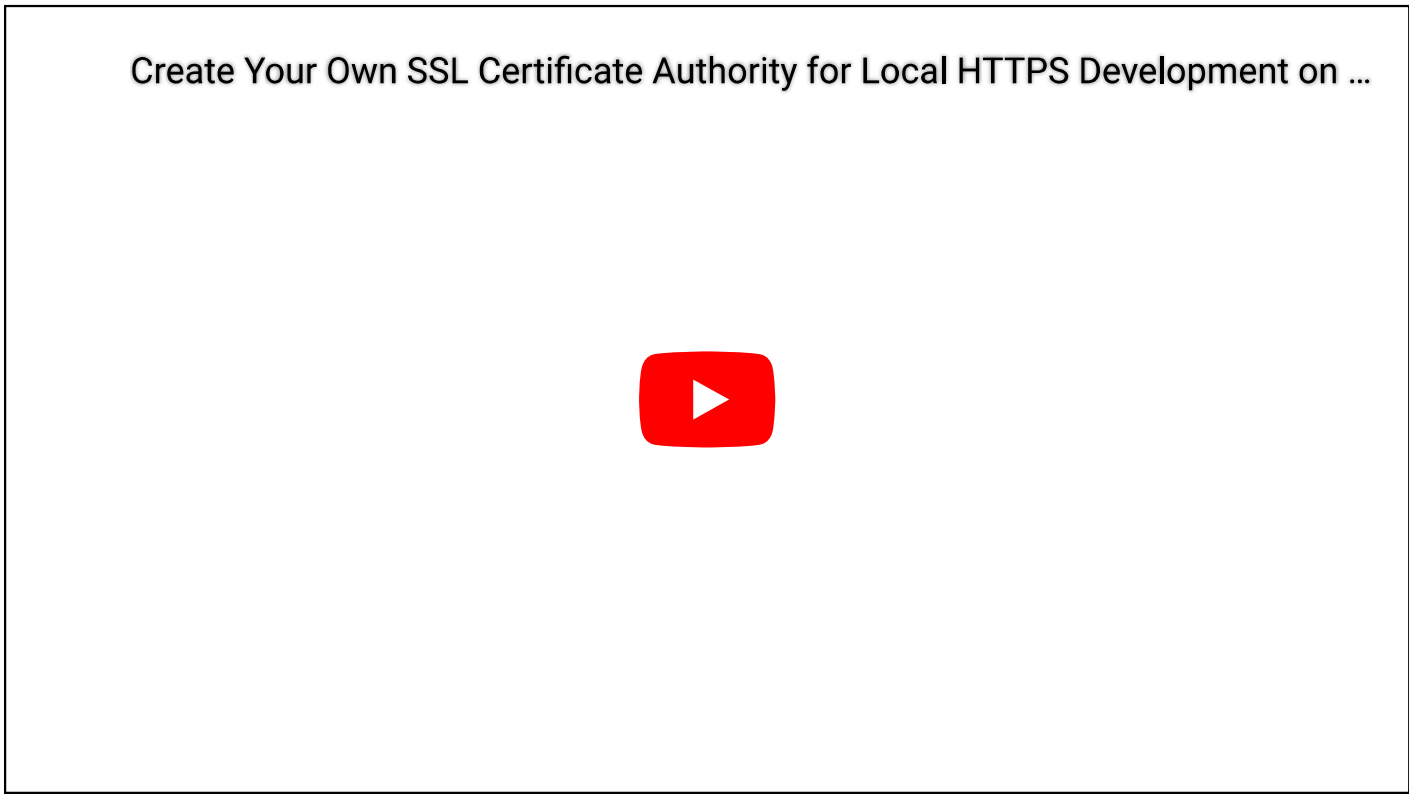
BY BRAD TOUESNARD, FOUNDER & CEO

In 2018 Google started advocating that sites adopt HTTPS encryption (https://security.googleblog.com/2018/02/a-secure-web-is-here-to-stay.html), by marking sites not using an SSL certificate as “not secure” in their Chrome browser. This was widely accepted as a good idea, as securing web traffic protects both the site owner and their customers.

While Let's Encrypt (https://deliciousbrains.com/http2-https-lets-encrypt-wordpress/) and its API has made it wonderfully easy for anyone to generate and install SSL certificates (https://spinupwp.com/hosting-wordpress-yourself-setting-up-sites/#obtaining-a-certificate) on their servers, it does little to help developers with HTTPS in their development environments. Creating a local SSL certificate to serve your development sites over HTTPS can be a tricky business. Even if you do manage to generate a self-signed certificate, you still end up with browser privacy errors.

In this article, we'll walk through creating your own certificate authority (CA) for your local servers so that you can run HTTPS sites locally without issue.

If you prefer to learn visually, our video producer Thomas has created a video for you that outlines the steps involved in creating your own local CA. He's also created videos that layout the process for Linux (https://deliciousbrains.com/videos/ssl-linux) and Windows (https://deliciousbrains.com/videos/ssl-windows) users.



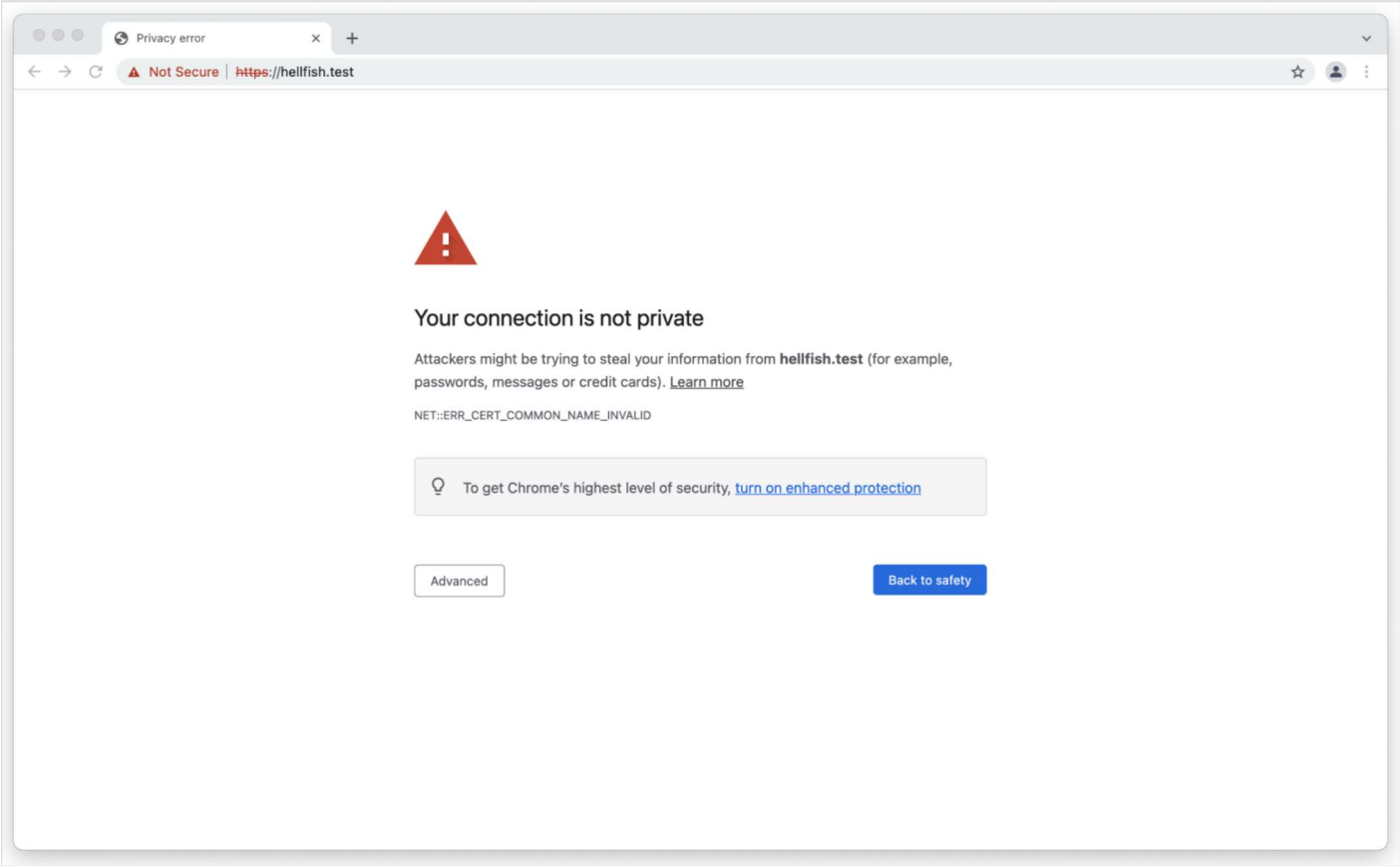
Why HTTPS Locally?

Why not just use regular HTTP locally? Because if your production site is HTTPS-only and you're developing locally on regular HTTP, your development and production environments are not as similar as they could be.

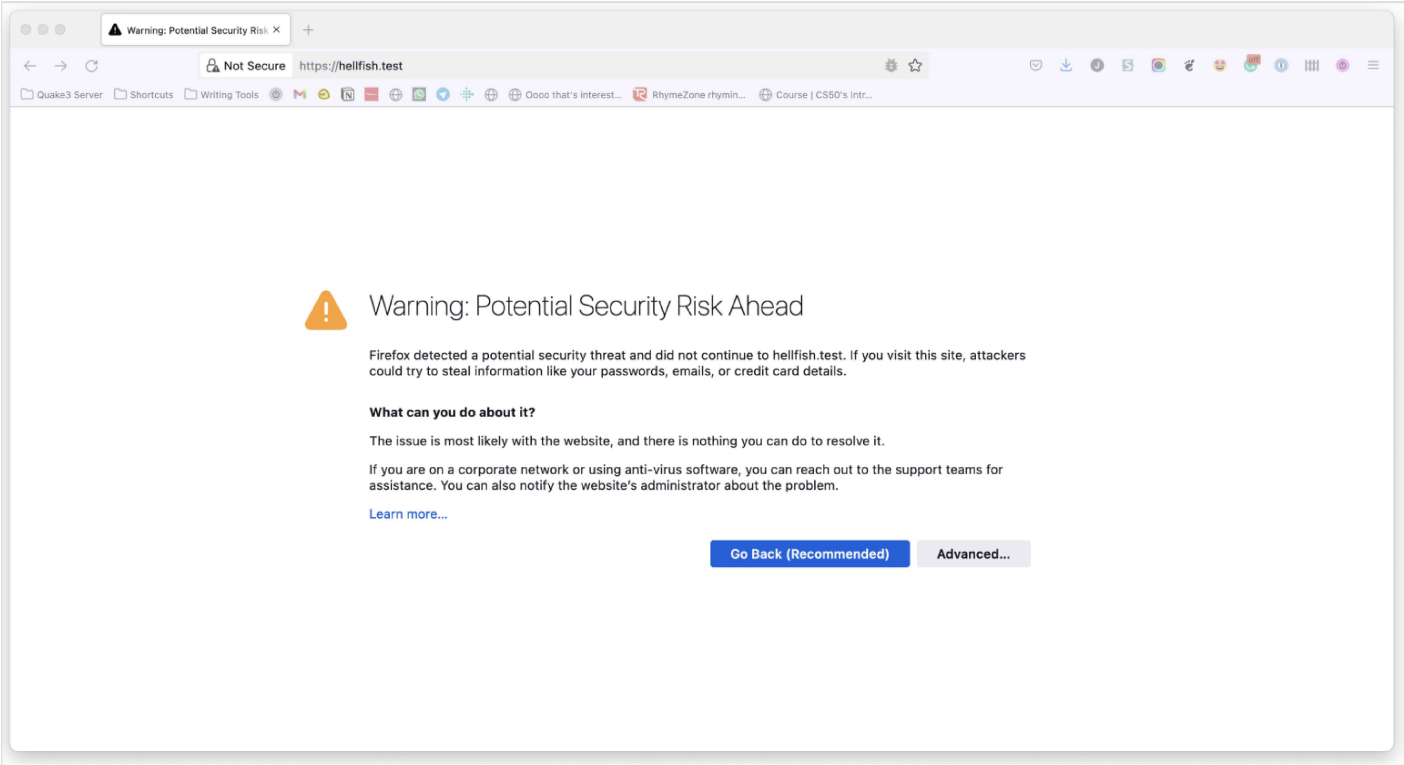
For example, my development environment for this site (https://deliciousbrains.com/) (and SpinupWP (https://spinupwp.com/)) runs as an Ubuntu server in a VMware virtual machine (VM) on my Mac. The production site is an Ubuntu server running on DigitalOcean (https://spinupwp.com/doc/getting-started/#provision-server) with an almost identical configuration.

You definitely want your development environment (https://deliciousbrains.com/xampp-mamp-local-dev/) to mirror production (https://spinupwp.com/hosting-wordpress-yourself-setting-up-sites/) as closely as possible. When it doesn't, you invite more issues showing up in production that didn't show up in development. Running HTTP when your production site is HTTPS-only is definitely an unnecessary risk. Even in a situation where you can't mirror your production environment perfectly, you'll still want to run HTTPS locally, or you'll be fighting with mixed content SSL warnings (https://wpbuffs.com/wordpress-https-mixed-content/) all day long.

If you've ever tried to browse to a local site via HTTPS, which doesn't have an SSL certificate configured, you've probably seen the following message in Chrome:



Or the following in Firefox:



Other browsers have different messages, but the gist is the same.

One way to work around this is to switch your local WordPress development environment to something like LocalWP (https://deliciousbrains.com/xampp-mamp-local-dev/), DevKinsta (https://deliciousbrains.com/craft-nitro-devkinsta-wordpress-local-development/), or even Laravel Valet (https://deliciousbrains.com/valet-vvv-chassis-comparison-to-cli-based-local-dev-environments/#valet) which offer local SSL solutions out of the box. The downside is that this means changing your development workflow, not ideal if you are more comfortable with what you already have, especially if it already matches your production environment.

Searching for a local SSL solution online will often result in you going down the rabbit hole of self-signed certificates. However, trying to get a self-signed SSL certificate working with your local server kind of sucks if you're not using a tool that handles it for you, which brings you back to needing to switch local development environments.

The main problem with locally self-signed certificates is that they also need to be trusted by your browser. Just setting up a local self-signed certificate isn't enough. You end up with the same browser message, but this time with `ERR_CERT_AUTHORITY_INVALID`. This happens because the browser wants to check the validity of this certificate with a certificate authority, and can't. So the solution is to become your own CA!

How It Works

To request an SSL certificate from a CA like Verisign or GoDaddy, you send them a Certificate Signing Request (CSR), and they give you an SSL certificate in return that they have signed using their root certificate and private key. All browsers have a copy (or access to a copy from the operating system) of the root certificate from the various CAs, so the browser can verify that your certificate was signed by a trusted CA.

That's why when you generate a self-signed certificate the browser doesn't trust it. It hasn't been signed by a CA. The way to get around this is to generate our own root certificate and private key. We then add the root certificate to all the devices we own just once, and then all the self-signed certificates we generate will be inherently trusted.

Becoming a (Tiny) Certificate Authority

It's kind of ridiculous how easy it is to generate the files needed to become a certificate authority. It really only takes two commands. Let's dive into how we can do this on macOS and Linux, and then look at how it works in the Windows operating system.

Generating the Private Key and Root Certificate on macOS Monterey and Linux

As macOS and Linux are both Unix-like operating systems, the processes for generating the required files are identical.

The only real difference between the two is that on macOS you might need to install the OpenSSL (https://www.openssl.org/) command-line application. To do that, if you don't already have it, install homebrew (https://brew.sh/), which will allow you to install OpenSSL.

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
brew install openssl
```

Copy

The majority of Linux distros come with OpenSSL installed. If not, it can be installed via your default package manager.

Then, we can create a location to store our local certificate files. This is not a requirement, but it makes it easier to find the keys later.

```
mkdir ~/certs
cd ~/certs
```

Copy

With that set up, we're ready to generate the private key to become a local CA:

```
openssl genrsa -des3 -out myCA.key 2048
```

Copy

OpenSSL will ask for a passphrase, which we recommend not skipping and keeping safe. The passphrase will prevent anyone who gets your private key from generating a root certificate of their own. The output should look like this:

```
Generating RSA private key, 2048 bit long modulus
.....................................................................+++
.....................................+++
e is 65537 (0x10001)
Enter pass phrase for myCA.key:
Verifying - Enter pass phrase for myCA.key:
```

Copy

Next, we generate a root certificate:

```
openssl req -x509 -new -nodes -key myCA.key -sha256 -days 1825 -out myCA.pem
```

Copy

You will be prompted for the passphrase of the private key you just chose and a bunch of questions. The answers to those questions aren't that important. They show up when looking at the certificate, which you will almost never do. I suggest making the Common Name something that you'll recognize as your root certificate in a list of other certificates. That's really the only thing that matters.

```
Enter pass phrase for myCA.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]: Springfield State
Locality Name (eg, city) []:Springfield
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Hellfish Media
Organizational Unit Name (eg, section) []:7G
Common Name (e.g. server FQDN or YOUR name) []:Hellfish Media
Email Address []:abraham@hellfish.media
```

Copy

You should now have two files: myCA.key (your private key) and myCA.pem (your root certificate).

🎉 Congratulations, you're now a CA. Sort of.

Generating the Private Key and Root Certificate on Windows

On Windows, it's also possible to configure your environment to run the `openssl` commands. You just need some additional tools.

If you are running Windows Subsystem for Linux (https://docs.microsoft.com/en-us/windows/wsl/about) (WSL) then it's as if you're running Linux and the commands will work exactly the same. If you're using Windows with something like WampServer or XAMPP you'll need a way to install the OpenSSL command-line utility in Windows. The most straightforward way to do this is to install Git for Windows (https://gitforwindows.org/), which comes bundled with OpenSSL and the Git Bash utility.

Once you open a Git Bash window, you can run the same commands as for macOS or Linux, with one small difference. Due to how some console applications (specifically OpenSSL) work in Git Bash, you need to prefix all `openssl` commands using the `winpty` utility.

So for example, the following command is how to generate the private key to become a local CA in Git Bash:

```
winpty openssl genrsa -des3 -out myCA.key 2048
```

Copy

The other small differences are the file paths in Git Bash. When you open a Git Bash instance, the home directory in the terminal is mapped to your User directory in Windows, but with a Linux-like directory structure. So if your User directory is located at `c:\Users\Hellfish` in Windows, your Git Bash home directory will be `c/Users/Hellfish` .

Installing Your Root Certificate

To become a real CA, you need to get your root certificate on all the devices in the world.

But we don't need to become a real CA. We just need to be a CA for the devices you own. We need to add the root certificate to any laptops, desktops, tablets, and phones that access your HTTPS sites. This can be a bit of a pain, but the good news is that we only have to do it once. Our root certificate will be good until it expires.

Adding the Root Certificate to macOS Monterey Keychain

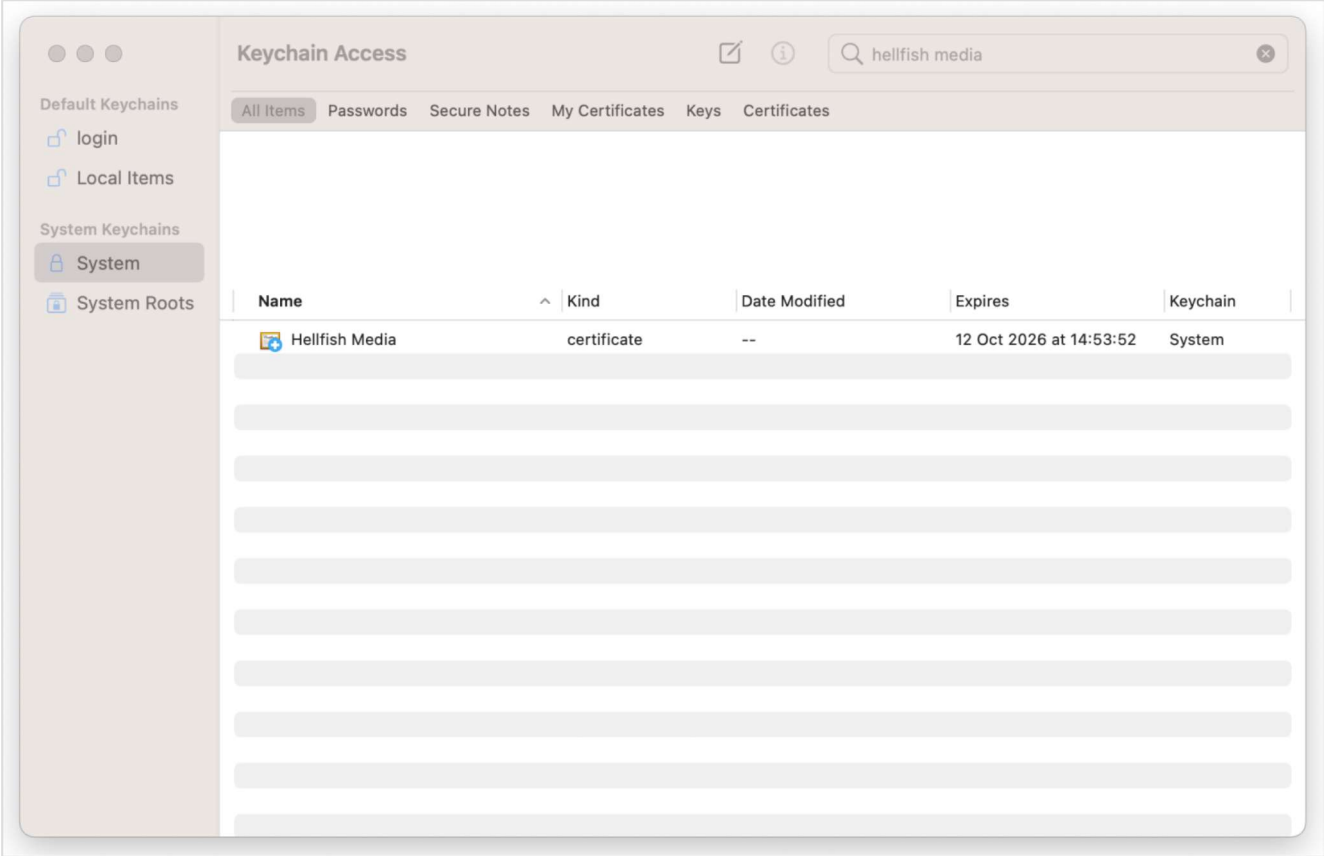
Via the CLI

```
sudo security add-trusted-cert -d -r trustRoot -k "/Library/Keychains/System.keychain" myCA.pem
```
Copy

Via the macOS Keychain App

1. Open the macOS Keychain app
2. If required, make sure you've selected the **System Keychain** (older macOS versions default to this keychain)
3. Go to **File > Import Items...**
4. Select your private key file (i.e. `myCA.pem`)

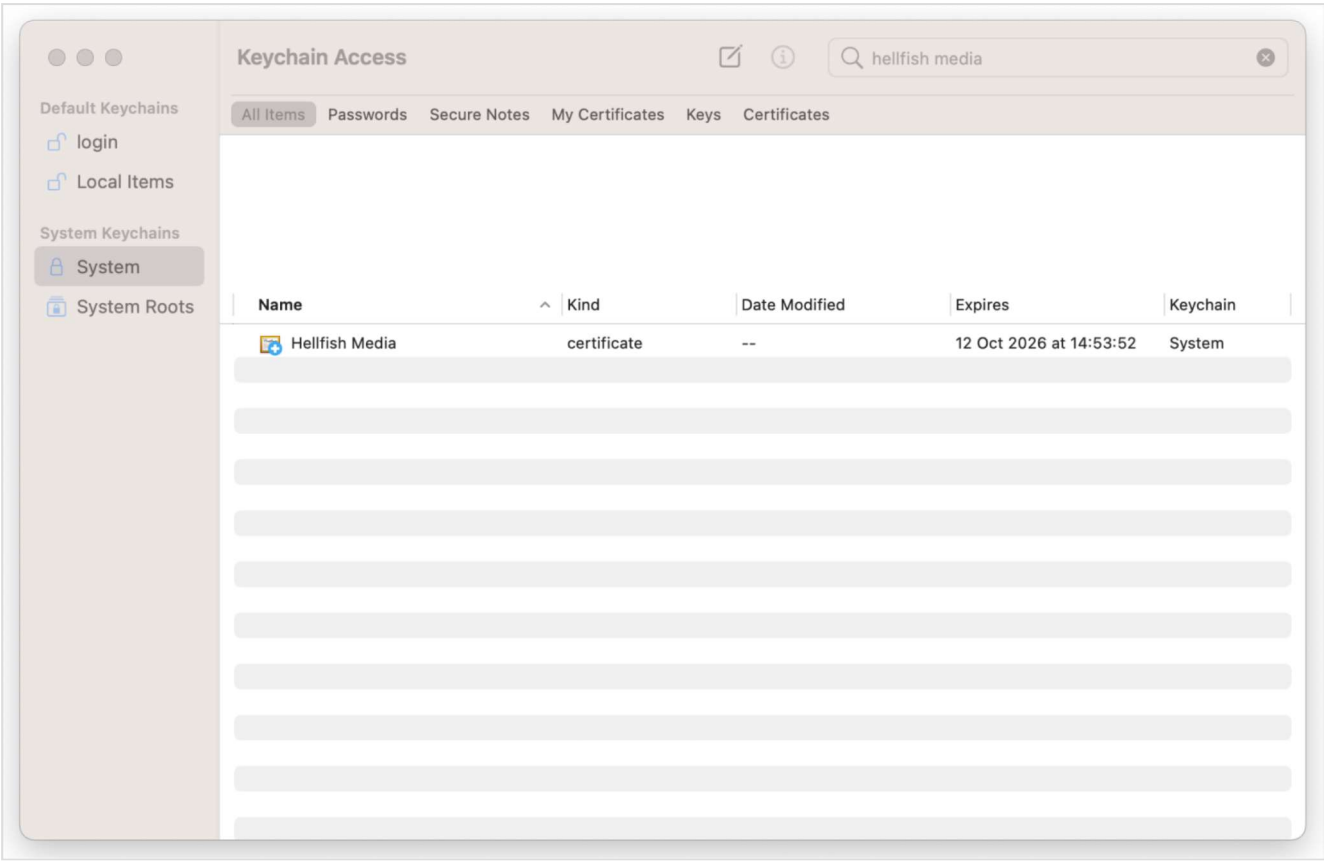
5 Search for whatever you answered as the “Common Name” name above



6 Double-click on your root certificate in the list

7 Expand the **Trust** section

8 Change the “When using this certificate:” select box to **Always Trust**



9. Close the certificate window
10. During the process it may ask you to enter your password (or scan your finger), do that
11. 🎉 Celebrate!

Adding the Root Certificate to Linux

There are so many Linux distributions, but Ubuntu is by far the most popular and it's what we used when we built SpinupWP (https://spinupwp.com/). Therefore these instructions will cover Ubuntu.

1. If it isn't already installed, install the `ca-certificates` package.

```
sudo apt-get install -y ca-certificates
```

2. Copy the `myCA.pem` file to the `/usr/local/share/ca-certificates` directory as a `myCA.crt` file.

```
sudo cp ~/certs/myCA.pem /usr/local/share/ca-certificates/myCA.crt
```

3. Update the certificate store. `sudo update-ca-certificates`

You can test that the certificate has been installed by running the following command:

```
awk -v cmd='openssl x509 -noout -subject' '/BEGIN/{close(cmd)};{print | cmd}' < /etc/ssl/certs/Copyerti
```

Copy

If it's installed correctly, you'll see the details of the root certificate.

```
subject=C = US, ST = Springfield State, L = Springfield, O = Hellfish Media, OU = 7G, CN = Hellfish Media, e
```

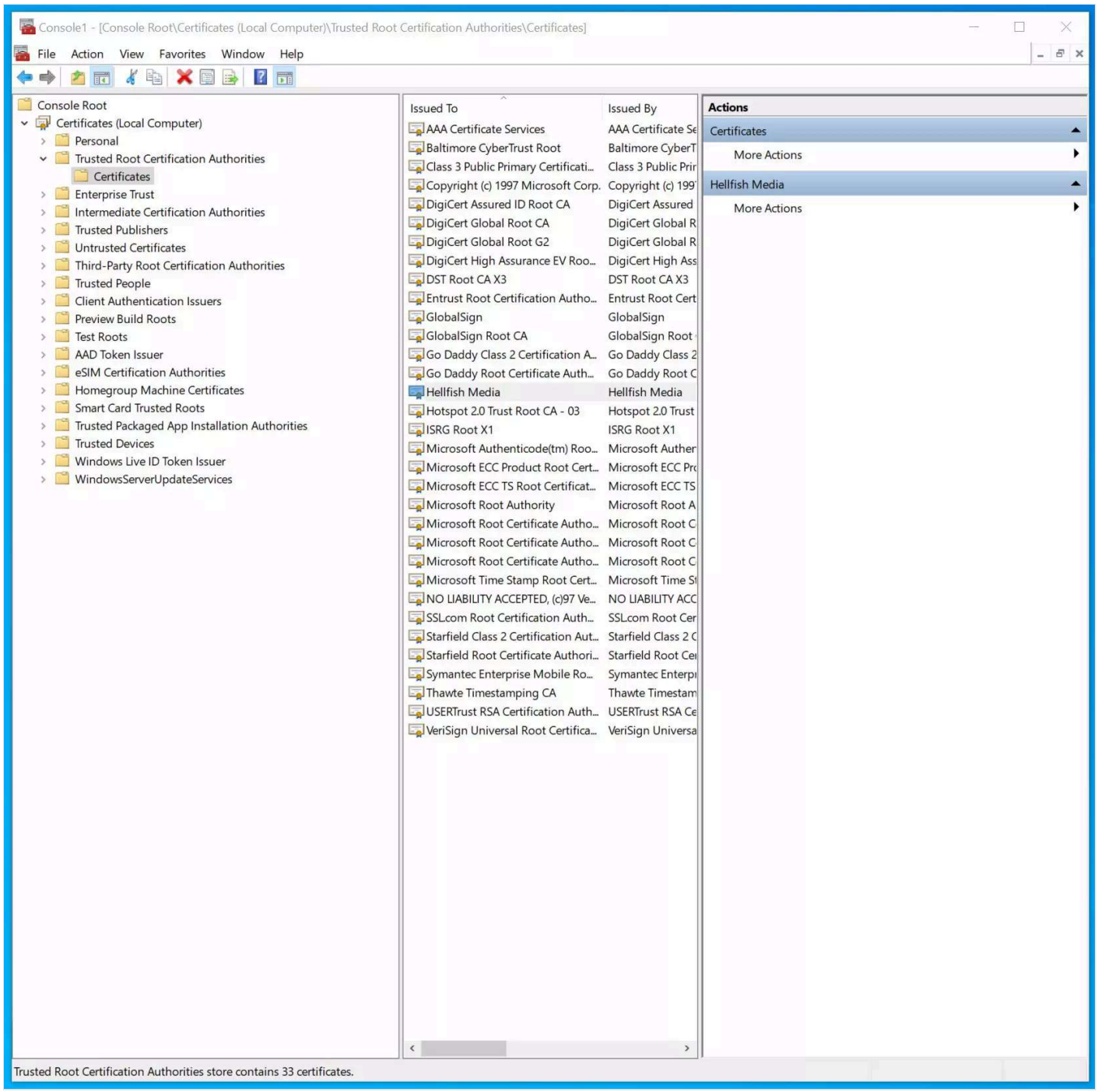
Copy

Adding the Root Certificate to Windows 10

1. Open the "Microsoft Management Console" by using the **Windows + R** keyboard combination, typing `mmc` and clicking **Open**
2. Go to **File > Add/Remove Snap-in**
3. Click **Certificates** and **Add**

4. Select **Computer Account** and click **Next**
5. Select **Local Computer** then click **Finish**
6. Click **OK** to go back to the MMC window
7. Double-click **Certificates (local computer)** to expand the view
8. Select **Trusted Root Certification Authorities**, right-click on **Certificates** in the middle column under “Object Type” and select **All Tasks** then **Import**
9. Click **Next** then **Browse**. Change the certificate extension dropdown next to the filename field to **All Files (*.*)** and locate the `myCA.pem` file, click **Open**, then **Next**
10. Select **Place all certificates in the following store**. “Trusted Root Certification Authorities store” is the default. Click **Next** then click **Finish** to complete the wizard.

If everything went according to plan, you should see your CA certificate listed under **Trusted Root Certification Authorities > Certificates**.

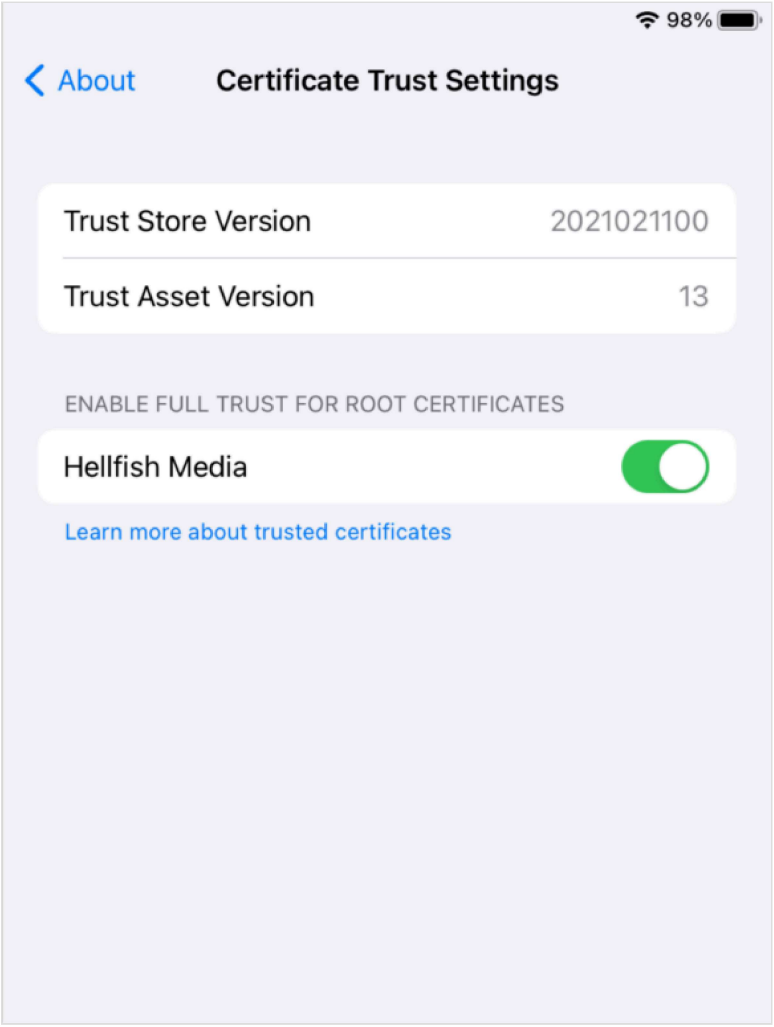


Adding the Root Certificate to iOS 14

If you use something like ngrok (https://ngrok.com/) to browse to your local development sites on mobile devices, you might need to add the root certificate to these devices. On iOS devices you can do so fairly easily by following these steps:

1. Email the root certificate to yourself, so you can access it on your iOS device. Make sure to use the default Mail app to access the email.
2. Tap on the attachment in the email on your iOS device. It will prompt you to review the profile in the Settings app.

3. Open the Settings app and click **Profile Downloaded** near the top.
4. Click **Install** in the top right, and then **Install** again on the Warning screen.
5. Once installed, hit **Close** and go back to the main Settings page.
6. Go to **General > About**.
7. Scroll to the bottom and click on **Certificate Trust Settings**.
8. Enable your root certificate under “ENABLE FULL TRUST FOR ROOT CERTIFICATES”.



Creating CA-Signed Certificates for Your Dev Sites

Now we’re a CA on all our devices and we can sign certificates for any new dev sites that need HTTPS. First, we create a private key for the dev site. Note that we name the private key using the domain name URL of the dev site. This is not required, but it makes it easier to manage if you have multiple sites:

```
openssl genrsa -out hellfish.test.key 2048
```
Copy

Then we create a CSR:

```
openssl req -new -key hellfish.test.key -out hellfish.test.csr
```
Copy

You'll get all the same questions as you did above and, again, your answers don't matter. In fact, they matter even *less* because you won't be looking at this certificate in a list next to others.

```
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]:Springfield State
Locality Name (eg, city) []:Springfield
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Hellfish Media
Organizational Unit Name (eg, section) []:7G
Common Name (e.g. server FQDN or YOUR name) []:Hellfish Media
Email Address []:asimpson@hellfish.media

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```
Copy

Finally, we'll create an X509 V3 certificate extension config (https://www.openssl.org/docs/man1.0.2/man1/x509.html) file, which is used to define the Subject Alternative Name (https://www.digicert.com/faq/subject-alternative-name.htm) (SAN) for the certificate. In our case, we'll create a configuration file called `hellfish.test.ext` containing the following text:

```
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
subjectAltName = @alt_names

[alt_names]
DNS.1 = hellfish.test
```

Copy

We'll be running `openssl x509` because the x509 command (https://www.openssl.org/docs/man1.0.2/man1/x509.html) allows us to edit certificate trust settings. In this case we're using it to sign the certificate in conjunction with the config file, which allows us to set the Subject Alternative Name. I originally found this answer on Stack Overflow (https://stackoverflow.com/a/43665244).

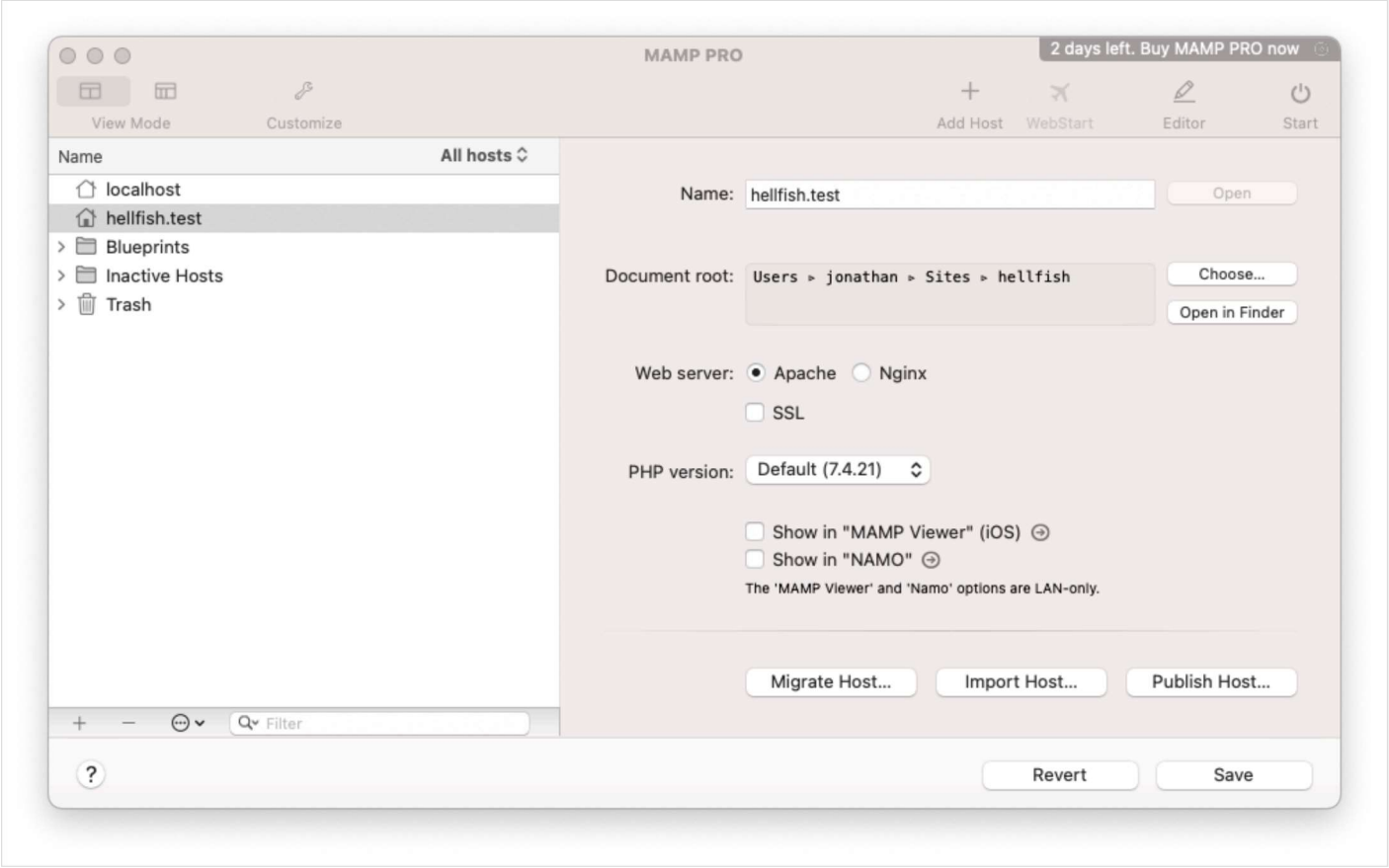
Now we run the command to create the certificate: using our CSR, the CA private key, the CA certificate, and the config file:

```
openssl x509 -req -in hellfish.test.csr -CA myCA.pem -CAkey myCA.key \
-CAcreateserial -out hellfish.test.crt -days 825 -sha256 -extfile hellfish.test.ext
```

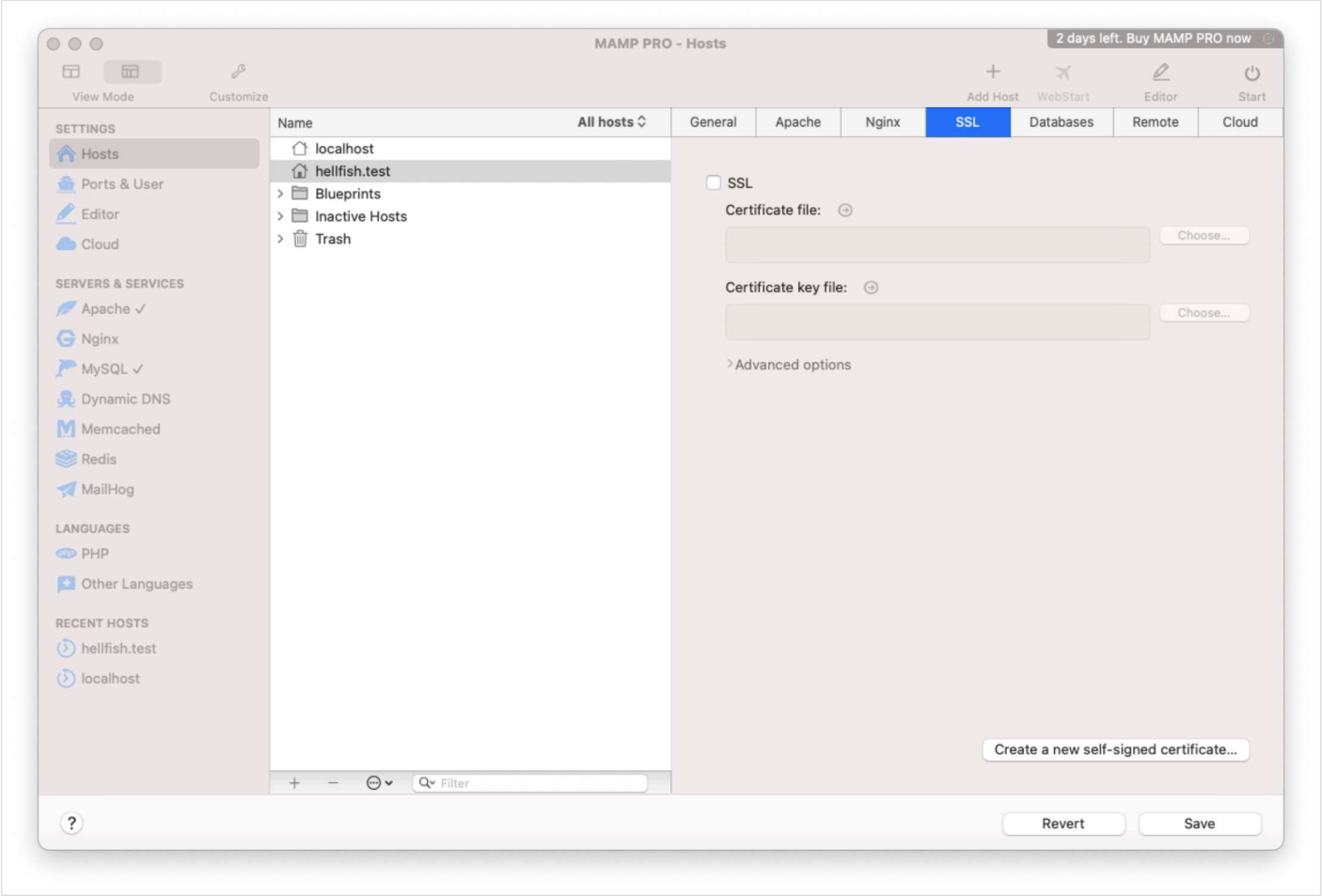
Copy

We now have three files: `hellfish.test.key` (the private key), `hellfish.test.csr` (the certificate signing request, or csr file), and `hellfish.test.crt` (the signed certificate). We can configure local web servers to use HTTPS with the private key and the signed certificate.

If you're using MAMP Pro, version 6.0 introduced built-in SSL support. You can enable it by checking the SSL box under your selected web server.



If you prefer to use the locally signed certificate we've just set up, you can do this by enabling the "Expert" view, clicking on the **SSL** tab, and choosing your "Certificate" and "Certificate key" (private key) files.



If you're running a Linux or Windows environment which uses Nginx you can use the instructions in our Install WordPress on Ubuntu 20.04 series (https://spinupwp.com/hosting-wordpress-yourself-setting-up-sites/#obtaining-a-certificate).

If you're on Linux or Windows using Apache, you'll need to enable the Apache SSL mod, and configure an Apache virtual host for port 443 for the local site. It will require you to add the `SSLEngine`, `SSLCertificateFile`, and `SSLCertificateKeyFile` directives, and point the last two to the certificate and key file you just created.

```
<VirtualHost *:443>
    ServerName hellfish.test
    DocumentRoot /var/www/hellfish-test

    SSLEngine on
    SSLCertificateFile /path/to/certs/hellfish.test.crt
    SSLCertificateKeyFile /path/to/certs/hellfish.test.key
</VirtualHost>
```

Copy

We don't have instructions for how to do this on Windows using IIS, because WordPress is not the easiest to configure on IIS systems.

We don't have to create a new CA for each site. We can just repeat this last part of creating a certificate for any other dev sites.

Shell Script

To make things even speedier, here's a handy shell script you can modify for your own purposes. It should work on macOS, Linux, or Windows via Git Bash:

Copy

```
#!/bin/sh

if [ "$#" -ne 1 ]
then
  echo "Usage: Must supply a domain"
  exit 1
fi

DOMAIN=$1

cd ~/certs

openssl genrsa -out $DOMAIN.key 2048
openssl req -new -key $DOMAIN.key -out $DOMAIN.csr

cat > $DOMAIN.ext << EOF
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
subjectAltName = @alt_names
[alt_names]
DNS.1 = $DOMAIN
EOF

openssl x509 -req -in $DOMAIN.csr -CA ../myCA.pem -CAkey ../myCA.key -CAcreateserial \
-out $DOMAIN.crt -days 825 -sha256 -extfile $DOMAIN.ext
```

Alternatives

An alternative to look into for creating locally trusted SSL certificates is mkcert (https://github.com/FiloSottile/mkcert) (thanks to the folks in the comments for pointing this out). If you don't mind using one of the various package managers listed in mkcert's readme file to install the tool, it's a solid alternative for creating locally trusted SSL certificates. The downside is that it only installs the root CA certificate for you on your local machine and creates locally signed SSL certificates, you still need to configure the certificates manually for each local site.

Conclusion

So there you have it, how to become your own local certificate authority to sign your local SSL certificates and use HTTPS on your local sites. Hopefully, this will eliminate the dreaded "Your connection is not private" message for you on your local development websites.

Have you tried setting up a CA of your own? Do you work locally with HTTPS? Let me know in the comments below.

This entry was posted in WP Migrate DB Pro (https://deliciousbrains.com/wp-migrate-db-pro/level-up/), Workflow (https://deliciousbrains.com/wp-migrate-db-pro/level-up/workflow/) and tagged SSL (https://deliciousbrains.com/tag/ssl/), HTTPS (https://deliciousbrains.com/tag/https/), Development Tips (https://deliciousbrains.com/tag/development-tips/), Development Environment (https://deliciousbrains.com/tag/development-environment/), MAMP (https://deliciousbrains.com/tag/mamp/), Certificate Authority (https://deliciousbrains.com/tag/certificate-authority/), OpenSSL (https://deliciousbrains.com/tag/openssl/).

ABOUT THE AUTHOR

Brad Touesnard, Founder & CEO

As founder of Delicious Brains Inc, Brad has worn many hats. He now spends most of his time managing the product teams and growing the business. Before starting this company, Brad was a freelance web developer, specializing in front-end development.

@bradt (https://twitter.com/bradt) bradt.ca (https://bradt.ca/)

94 comments

Oldest

Sign in to join the discussion

ReplyBox Google

Anton VS

Is there any reason to set up an SSL certificate / HTTPS for local development?

1

Reply Upvote

Henrique Mattos

Developers have been editing computer hosts file to redirect the original domain (say example.com) to localhost (say 127.0.0.1) so they can use the fully qualified URI/URL in the development. It's a good way to develop WordPress themes and plugins and then upload those to the production webserver not needing to script into the DB to rewrite permalinks, attachment URLs, etc... Also, having HTTPS is mandatory for some WooCommerce plugins or some XSS integration and therefore it's nice to have it in your dev environment. :)

Reply Upvote

Laukik Patel

Thank you so much @bradt

Reply Upvote

numediaweb

Thanks so much! I turned this into an Ansible role which allows me to generate unlimited hosts with each one a unique cert!

Reply Upvote

gmannz

Thanks Brad, this was a good concise article and worked well. I secured my WIFI AirOS nano WIFI AP's with a new certificate, as well for my lab I will be applying these to some other devices. Keep up the good work. Greg

Reply Upvote

Clément Guinet

Thanks for this very useful post :)

Reply Upvote

Tanner

Genius! I used this tutorial to help with local Traefik & docker. Thanks for making it rather easy to follow.

Reply Upvote

Dev

Is it possible to issue a Wildcard? I've tried setting common name as *.mydoman.com but I get ERR_CERT_COMMON_NAME_INVALID from chrome. BTW many thanks for the useful article!

1

Reply Upvote

Thomas Bacon

Yes it is, but as mentioned in this article: https://deliciousbrains.com/https-locally-without-browser-privacy-errors/ setting the common name is insufficient, you have to set it in the SAN Config file. For example: `DNS.1 = *.domain.dev` As a matter of fact I set this up so that I can use it for the purpose of making it super easy to setup local HTTPS. I just use the format of my-site.domain.dev, my-site-2.domain.dev, etc...

Reply Upvote

Benjamín Burgos

Geat article. Works like a charm. Thanks a lot!

Reply Upvote

EugeneX

Thanks a lot! Very nice post!

Reply Upvote

dobes_vandermeer

I put this all together in a shell script you can run:
https://gist.github.com/dobesv/13d4cb3cbd0fc4710fa55f89d1ef69be

Reply Upvote

Martin Reinke

I tried to get this working on Windows 10 the last two days. I could see, that the public key and the serial no in the certificate received by the browser was different from key and serial no produced by openssl. Only Firefox received the right key. In the end I found out, that the AVG Online Shield had manipulated part of the certificate and made it useless that way. After switching off the SSL trafic scan in AVG everything worked as it should. So keep your AV-Software in mind, when it is not working.

Reply Upvote

Keno Clayton

Thank you so much. After so many attempts with other articles I finally found success with yours
3 https://uploads.disquscdn.com/images/8fc70b87890c60e3e36246771017cd7b7528bfe708541dd2

Reply Upvote

Enk

Hello, can you tell me how you did it. I have wasted many hours trying to get by the NET::ERR_CERT_COMMON_NAME_INVALID on Chrome. I access my local at https://192.168.7.13/myapp and I set the DNS1 = myapp.domain.com but it doesn't seems to work. Thanks.

Reply Upvote

Keno Clayton

Also why did you set your DNS1 to be `myapp.domain.com` ? Your local server is `192.168.7.13` so I'd expect that to be your DNS1. Correct me if I'm mistaken.

Reply Upvote

Keno Clayton

Hi, just saw your reply. All I did was follow the steps in the tutorial. Make sure you follow this part as it deals with defining the Subject Alternative Name (SAN) which is needed to fix the error you're having. Let me know how it goes.

Reply Upvote

Clinton Brits

Can you recommend an article on the basics of ssl itself? I have always used a self signed cert to to my sites and just ignore the warnings. I now want to implement a windows tcp app that uses ssl. There is provision for key file, cert file, and root cert. From your article i can get all 3 but im confused as to what goes where? Does the cert and key reside on the server side application and the root cert in the client application? On one article they say a normal cert authority's root cert is added to new releases of browsers and then they say they are closely guarded?

Reply Upvote

Felix Ortego Perez

thanks a lot for the explanation

Reply Upvote

kingkool68

I see others have shared shell scripts that incorporates the commands in this article. So here's my take https://github.com/kingkool68/generate-ssl-certs-for-local-development If you're on a

Mac it automatically copies the root certificate to Keychain saving you a step.

Reply Upvote

Lakshmeepathi

Thanks. Wonderful article. I have a question. I was under the impression that only the private key of the CA is used to sign (sign our CSR / Public Key). But here both the Private Key of CA and CA's Public Certificate (Root Certificate) is used. Can it be further explained why both are needed in a simple manner or can it be understood only with the knowledge of cryptography ? Also why are you loading Private Key into KeyChain Access - in the article "Select your private key file (i.e. myCA.pem)"? Did you actually mean the CA's certificate file ?

Reply Upvote

Lakshmeepathi

Shouldn't the mentioning of SAN be done at the step of CSR creation as that seems more intuitive and appropriate - since CSR is the "request" shouldn't it mention for what CN/SAN it wants the signature for?

Reply Upvote

Ryan Liu

Note that once you create a serial using the CAcreateserial you can use the serial again: openssl x509 -req -in dev.mergebot.com -CA myCA.pem -CAkey myCA.key -CAserial myCA.srl -days 1825 -extfile dev.mergebot.com.ext -out dev.mergebot.com.crt

Reply Upvote

nomailme

Have been there, so I've created small test CA project: https://github.com/nomailme/TestAuthority It allows to issue test SSL certificates via REST API (or Swagger UI if you prefer). You can compile it and run in Win/Linux or as I prefer docker container

Reply Upvote

Enk

https://uploads.disquscdn.com/images/12debafac146b971b4e188f60fcc873ea6c0a4fbdae967eef I am not sure what I did wrong, but I've tried almost everything and still got the NET::ERR_CERT_COMMON_NAME_INVALID error with the message "This server could not prove t its security certificate is from kb.dci.com". My .ext is exactly the same as the article with the follow DNS.1 = kb.dci.com DNS.2 = kb.dci.com.192.168.7.101.xip.io I am on CentOS 7 and my hostname all the keys and certs in a custom directory (/etc/httpd/pki) and updated the ssl.cnf accordingly. A be appreciated. Thanks!

Reply Upvote

Tony Pedley

Nice article. Just to add a comment or two. It would be nice to add the SAN to the CSR, but there does not seem to be a valid way of doing it, so it has to go into the CA request. The biggest reason for us to become a CA, is that we are talking to embedded controllers that do not have a FQDN, only IP addresses. Regular CA's will not generate a certificate for anything other than a domain name. As the CA we can generate a SAN with multiple IP addresses (IE for some reason demands the IP addresses to be DNS values, heh ho). Biggest issue as acting as your own CA, is security and certificate management i.s managing CRL, however for a local intranet, these area manageable

Reply Upvote

themaster

In Case I need to create a signed certificate for my locahost:port. Would I have to change the openssl genrsa -out dev.mergebot.com.key 2048 to openssl genrsa -out dev.localhost:8800.key 2048 ??

1

Reply Upvote

MikeSchinkel

ports don't matter fyi it's just the parent dns record

Reply Upvote

Bobby Gecko

I recently attempted this setup and tried the steps outlined in both this post as well as numerous others - alas I had no success. I also tried TinyCA and RCA but both were really outdated and pretty much unusable. It took me a while but I finally found a reasonably well-made (and free) PKI management program (multi-platform) that uses a web interface so it's considerably easier to use than openSSL via the command line (from what I understand however, the application does actually use openSSL underneath - so you could think of it as a front-end for openSSL). I did a breakdown on TLS basics as well as some tips for using the aforementioned tool on my blog at the link below. I hope this is as helpful for others as it was for me, now I have to go: there's a moth in the room that's about to get it... https://www.tech-jungle.com/setup-your-own-tls-certificate-authority-in-lieu-of-self-signed-certificates/

Reply Upvote

arvydasj

Important: if you want your CA certificate to work on Android properly, then add the following options when generating CA: openssl req -x509 -new -nodes -key myCA.key -sha256 -days 1825 -out myCA.pem -reqexts v3_req -extensions v3_ca

Reply Upvote

Phillipp

When I import it on android, it shows up as an user certificate and not as a CA certificate. It also doesn't show up under trusted access. Any tips on how to get it working?

2

doesn't show up under trusted access. Any tips on how to get it working?

Reply Upvote

Abidul Ramadan

you need to add the CA one (first one you generate) not the second one.

Reply Upvote

Abidul Ramadan

you need to add the CA one (first one you generate) not the second one.

Reply Upvote

B.R.

Perfect! Thank you so much!

Reply Upvote

Elliott Vernon

I have managed to create my own TLS certs using bare, arcane OpenSSL commands, with much help from https://jamielinux.com/docs/openssl-certificate-authority/. It's pretty torturous. I read in the OpenSSL documentation that these commands were never intended as much more than a proof-of-concept, but people seem to be using them for real because HTTPS everywhere is the future. LetsEncrypt is great but you can't use it on a private intranet, so... do we have much other choice?

Reply Upvote

Lyas Spiehler

https://certificatetools.com makes this very simple and generates the OpenSSL commands you can use to do it offline.

Reply Upvote

Firmwaree

Hello, thansk for this tuto ! i created a self signed certificate for my internal load balancer ! i try to add it to aws acm but i still get this error "An error occurred (ValidationException) when calling the ImportCertificate operation: com.amazonaws.pki.acm.exceptions.external.ValidationException: Provided certificate is not a valid self signed. Please provide either a valid self-signed certificate or certificate chain." even if i convert the cert and his key in pem format i still get the same error ! now i believe because it signed with my authority i need to provide a certificate chain ! How can i do it ? thanks

Reply Upvote

Anders Rørvik

Thanks so much for the tutorial Brad!

Reply Upvote

haptomatic

Thanks for this guide, it's been a huge help!! Everything was working fine until I formatted the Mac I generated everything from today. All I've done since then was import and trust the Root CA again in Keychain Access. Now when I visit something in Chrome, it will definitely find the certificate, but it says it's been revoked. It's weird though, because I remember specifically trusting the Root CA on an entirely different computer than the one I generated it from, in order to test it originally, and everything was fine. Will have to investigate that later to see if it still works. https://ibb.co/yh76z2B

Reply Upvote

Allen Bradley

Thanks for the tutorial. I used the instructions to create a private key, cert, and ca to connect from Celery container to Redis container as required in hereBut I have problems to connect. Can I use certs that were generated in one environment in another environment? For example, I created the certs in localhost. Can I use them to connect from a Celery docker container to a Redis docker container? My specific question with more details is posted hereThanks

Reply Upvote

joshua ashley

Great article my man

Reply Upvote

Hike Nalbandyan

This was super helpfull!! Thank you

Reply Upvote

nydame

Nice article. Next question, is there any way to distribute CA's root cert to all windows machine joining the same domain? So we don't have to install the root CA's cert manually one-by-one.

Reply Upvote

jvanpelt

I just use ngrok, I know you can roll your own but it just works and that's worth paying the annual fee for. Their tool that lets you inspect all traffic that goes through it is also great.

Reply Upvote

Hannes

just noticed that `.srl` file in the directory where i signed my Certificate Signing Request (CSR). the web told me this file contains a serial key that i need to provide to any other certificate signed with the same Certificate Authority (CA). i should do that with `-CAserial ca.srl` is that

signed with the same Certificate Authority (CA). I should do that with `-CAserial ca.srl`. Is that correct? if so, it might be nice to add. thanks you for that well guided tutorial! hannes

source: http://www.gutizz.com/openssl-creates-ca-serial-file/

Reply Upvote

Dave Varon

FInally a tutorial that renders a successful result: Mac OS X Catalina, Vue cli 3.x, Webpack Dev Server, Chrome 87.blah.blah

Reply Upvote

webaware

This is something that I've been doing for ages, but when I mentioned it on a Slack channel a security expert told me how this could be used to MITM attack me if the CA cert keys were stolen. Pretty low risk, but huge impact if it happened -- say hello to successful expert phishing attacks.

Apparently the way to fix this is by adding Name Constraints to the CA cert, restricting the domains that it can apply to. Here's two discussions on how.

https://security.stackexchange.com/a/130674/218836
https://systemoverlord.com/2020/06/14/private-ca-with-x-509-name-constraints.html

Reply Upvote

Daniel Seuffer

Finally my local certificates are working again. Thank you very much for this great post. I got stuck for some hours and walked through 4 other explanations before i ended up here. This morning i've encountered some cors issues because of cross domain session/cookie usage and so i had to solve my local ssl issues before i can go on. Here you can find my email (https://github.com/authanram), if you send me your paypal addy a donation link smth. similar, i will send you a few bucks. Thanks a lot.

Reply Upvote

Aleksandr Filatov RU NL US

Shared my knoweldge about similar issue in my post here https://alfilatov.com/posts/how-to-create-self-signed-certificate/

Reply Upvote

drewrunolfsson

Thanks Bro , Recently added SSL on Client site https://migweldertech.com , You save my time.

Reply Upvote

Mehmet Gültekin

Awesome. Thank you.

Reply Upvote

Mohammad Hasan Mia

Does anyone know how to generate self signed root certificate on Win 10 for Xaomi router using openssl (I would like to send all traffic using ssl to router)

Reply Upvote

catalin detest

Exactly awesome.

Reply Upvote

Volkan Kaban

Do you prefer to install SSL your way or to use mkcert? Somehow we are sharing our information with 3rd party. When we pay for SSL we are comfortable because it's business. Why Firefox or chrome it's not providing free local SSL for developers? Because nothing it's free?

Reply Upvote

Juan Mamani

Really useful! Thanks for sharing.

Reply Upvote

Lior Talmor

I'm using devilbox for my local development. It's absolutely perfect and it also takes care of the local ssl certificate / https local domain. There's an article talking about it as well on the Delicious Brains blog :-) , you could list it as an alternative and link to the post ;)

Reply Upvote

A Mobile Way

Hi, In the final third step I tried to run openssl x509 -req -in localhost.csr -CA myCA.pem -CAkey myCA.key \ -CAcreateserial -out localhost.crt -days 825 -sha256 -extfile localhost.ext on my Mac and it gave an error - unknown option -CAcreateserial yet it also listed - usage: x509 args with all options including -req - input is a certificate request, sign and output. -CA arg - set the CA certificate, must be PEM format. -CAkey arg - set the CA key, must be PEM format missing, it is assumed to be in the CA file. -CAcreateserial - create serial number file if it does not exist -CAserial arg - serial file -set_serial - serial number to use -text - print the certificate in text form

So what could be wrong?...

cheers

Reply Upvote

Matthew Bowen

Thanks so much! This guide was incredibly helpful.

Reply Upvote

rosario moggia

Hi. Your tutorial is very interested. I followed him. But when I connect to the Apache webserver, it tells me that the certificate is not valid because it cannot contact the CA

Reply Upvote

Ross Moore

When I follow the steps in this article I am able to create all necessary certificates. I have added the Root certificate to the Trusted Roots store. However, when I use openssl to combine the public and private keys to a pfx, and then import that pfx to an IIS server, I get the error of Err_cert_authority_invalid. I used this command: openssl pkcs12 -export -out certificate.pfx -inkey privateKey.key -in certificate.pem. Do I also need to add the private root certificate in the pfx? And if so, how do I do that?

Reply Upvote

Deepak Kumar

Getting error "Command Not Found" while typing

keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment

Please help

Reply Upvote

Walt Williams

I followed your article but couldn't get a local, private CA to work. I will go through it again. Perhaps I missed something.

My question is: I am trying to set up a private CA for development purposes on the same computer the browser running on. Will this need to be on be on a separate computer? Perhaps a Linux Container or Virtual-box with Debian 10? I also have a Raspberry PI 2B I could use.

Thanks in advance Walt Williams

Reply Upvote

Olakunle Oni

Hello guys! Please I don't understand how you guys did it, when adding the root certificate to Linux, I expected 2 errors and I got them. 1. This directory does not exist /usr/local/share/ca-certificates, how did you copy the cert to the directory cos I know the directory does not truly exist 2. sudo-update-ca-certificates gave me a command not found error and finally awk -v cmd='openssl x509 -noout -subject' '/BEGIN/{close(cmd)};{print | cmd}' < /etc/ssl/certs/ca-

certificates.crt | grep Livetest threw an error of /etc/ssl/certs does not exist. Please point me in the right direction. Thanks.

Reply Upvote

Andrii Tykhonov

Hi! Thank you very much for the article. I found it after some experience with mkcert. Yes, mkcert is helpful, but also very helpful to understand what happens behind the curtain (in order to be able to solve an issue, if any). And the article helped me a lot. Thank you!

Reply Upvote

k

kuo eric

Thank you so much. It's working.

Reply Upvote

Antonio Nieto García

Thanks a lot Brad! Very useful and well explained tutorial

Reply Upvote

Dennis Barzen

Great article, Brad. Thank you very much.

There is one thing I stumbled across today when I was testing a dev environment with iPhone Simulator and Safari told me that the connection was not secure. Since September 2020, certificates are allowed to be valid for a maximum of 398 days. https://support.apple.com/en-us/HT211025

Reply Upvote

1

W

WebDev

"This change will not affect certificates issued from user-added or administrator-added Root CAs"

Reply Upvote

Philip Rudy

For the windows tutorial, when I doble click "Certificates (local computer)" I dont' get the following options listed in your directions I get a prompt with two options:

1) Always enabel all available extentions 2) Enable only the slected extentions

Reply Upvote

Solo90

Hi, Thanks, but what a nightmare. I spent an hour mistyping the CN/OU etc. It's hell. Worse, the ^

does not work on Openssl.

You really need to add something along the lines of : -subj "/emailAddress=abc@example.eu/C=EU/ST=BE/L=Brussels/O=example/OU=MSS/CN=example.

and also add the email somewhere with : /emailAddress=webmaster@example.com But this mig now be done with subjectAltName.

P.S Why did you put in cd ~/certs. This serves no purpose other than generate an error. Maybe y could add [-d ~/certs] || mkdir -p ~/certs && cd ~/certs but it's just not required.

Any way, few are masicistic enough to endlessly enter typos into openssl commands.

Reply **Upvote**

ramon

i am soooo graeful for this post it saved me so much time! I thank you! I missed the part where the site needs to be added to hosts file though.

Reply **Upvote**

Daniel Trelogan

Great article, but on Ubuntu, update-ca-certificates activated the root cert in curl and wget, but not in my browser. To enable the new root cert in your browser:

Chromium: chrome://settings/certificates -> authorities tab -> import Firefox: Preferences -> Privacy & Security -> Certificates -> View Certificates -> Import

Reply **Upvote**

Maqué _

Thank you for the guide So happy to be a CA myself 😁

Reply **Upvote**

Geoff Vane

This is beyond comprehension for a stupid consumer. My NAS supplier says it's best to pick non-standard port numbers. But then the NAS is no longer trusted. This can be corrected by downloading a crt and key. The NAS can get them. Alas, it doesn't accept it's own downloaded crt and key. It's beyond me why this has to be such a nerd thing: just let me click somewhere to

SIGN UP FOR THE LATEST PRODUCT NEWS AND UPDATES

Enter your email address

SUBSCRIBE

Products

Advanced Custom Fields (https://advancedcustomfields.com) (https://twitter.com/wp_acf)

WP Migrate (https://deliciousbrains.com/wp-migrate-db-pro/) (https://twitter.com/wpmigratedbpro)

WP Offload Media (https://deliciousbrains.com/wp-offload-media/) (https://twitter.com/wpoffloadmedia)

WP Offload SES (https://deliciousbrains.com/wp-offload-ses/) (https://twitter.com/wpoffloadses)

Explore

Home (https://deliciousbrains.com/)

Blog (https://deliciousbrains.com/blog/)

Podcast (https://podcast.deliciousbrains.com/)

Contact Us (https://deliciousbrains.com/contact-us/)

My Account (https://deliciousbrains.com/my-account/)

Company

About (https://deliciousbrains.com/about/)

Careers (https://deliciousbrains.com/about/#careers)

Press (https://deliciousbrains.com/press/)

Giving Back (https://deliciousbrains.com/about/#giving-back)

(https://deliciousbrains.com/) **Delicious Brains Inc.** (https://deliciousbrains.com/)

@dliciousbrains (https://twitter.com/dliciousbrains)