



# Creating your own SSL Certificate

This document briefly explains how to create your own SSL Certificate for use in a server. This allows you to use a secure connection to communicate with your server.

SSL Certificates are based on 'Trust'. "who do you trust" to guarantee that no one has any access to your root certificate, that would enable someone to 'clone' or 'duplicate' a certificate, or create a new certificate using stolen credentials.

It's all based on Risk Management. "How far are you willing to go to ensure security".

For some home owners, it means putting a lock on the front door. For other homeowners, it means putting 5 deadbolts on the door and a fancy security system.

Each 'entity' (personal or professional) should have a 'security policy' that determines what they can/should do to prevent a catastrophic breach of security.

For most of us, it means putting an SSL Certificate on your server to ensure secure server/client communications.

Commercial SSL certs cost between \$50-100 US Dollars. Sometimes more, occasionally less. But for a hobbyist, or a low end provider, sometimes security isn't as critical as low cost. In these instances, there is nothing to keep you from creating your own certificate.

Here's how.

Oh, and although I'm talking about 'web' SSL for the most part, the underlying technology applies for email and other protocols. A certificate is issued to a particular server FQDN. There is nothing holding you back from running multiple services securely as long as the FQDN matches.

Note: I work with Mac, Windows, and Linux. This particular tutorial will be working in a Windows 7 environment. If you're using a Mac or a Linux machine, the steps will be 'similar' but not always the same. Caution should be excersized to ensure correct steps are taken regardless of your environment. Other windows versions will be very similar, with some differences in 'location' due to subtle differences in Windows Versions. Don't be afraid to stop and google something particular to your environment.



## Excuse me, what in the heck are we talking about?

Once upon a time, when the internet was young, people shared ideas and information. They gave of themselves freely, and a great void was being filled at an astronomical rate. Then came spammers, hackers and viruses, and other man-made chaotic occurrences, and people had to start tightening up their resources to keep from being abused. Data sharing became much more cumbersome. And now 'real' information is a lot harder to come by. You typically have to register somewhere, and provide credentials. At minimum a username and password. Sometimes financial information.

In the old days, there wasn't much security, because there wasn't much need. Now there is. A lot. Before, a hacker could 'listen' to a server port, and siphon all the data from any requests and transactions. Later, they would sift through the data for usernames and passwords. For unsecured sites, this still presents a security issue. So they developed 'SSL', secure socket layers. In a nutshell, a protective 'tunnel' between your client and the server.

What happens in a basic sense is, the server will determine if the requested address is http (regular web traffic) or https (secure web traffic). Typically what occurs is the server will 'redirect' all requests to https. Not always, but usually. It's safer that way. But I digress...

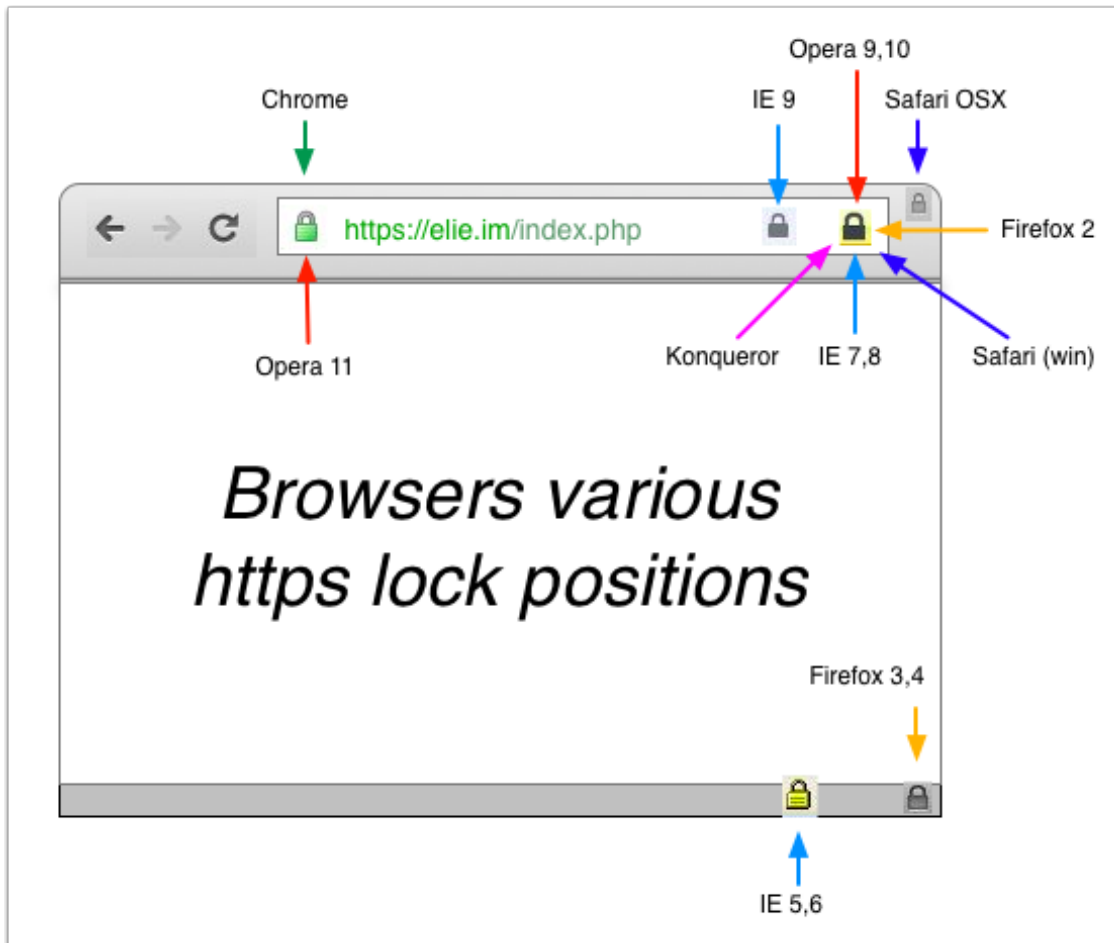
So, the server gets an https request, or a 'secure web page request'. In this example, let's say it's a login page. Without SSL, your login name and password would be sent 'in the clear'. With SSL, an encrypted 'tunnel' is created, and any data travelling to and from the server will be encrypted and protected by that tunnel. So your login name and password are protected against any 'sniffers' trying to break into a system.

"Usually" but not always, a site SSL Certificate will have been issued by an entity that has already been placed in your root store. In fact, almost all of them have. otherwise you would get alerts all the time.

The SSL Certificate we create will not have gone through all the vetting that a commercial provider must. Thus, our certificates will not be in the root store. Every time a new user visits our site (or a returning user who chose not to trust our cert) will get an alert notifying them of an issue with the certificate. "The Issue" is that the Root Server (the one we're going to create) is not in the Trusted Root Store. They will have an option to add the certificate, or not. Sometimes (depending on the browser) they can continue browsing, but the certificate is not stored, and each visit will bring a new alert dialog.



# Creating your own SSL Certificate





## A Quick word about 'Ports'

A server 'Port' is a network connection for services. There are thousands of ports available to a server. Ports are organized and assigned for the most part. Those that have official 'assignments' are called 'standard ports'. Those that do not have a designation are called 'non-standard ports'. There are far more 'non-standard' ports available than have been assigned.

Think of a port as a 'door' on a building with thousands of doors, all with the same street address (the server IP) but each with a unique apartment number (port number). SMTP (Email send) 'lives' behind door 25. FTP lives behind door 21. Standard Web (HTTP) lives behind door 80. Pop3 (email receive) lives behind door 110. And so many more... but Secure Web Traffic (HTTPS) lives behind door 443.





## Certificates. Theirs, yours. Trust.

So how does all this work? Well, the server has a certificate that states that it can provide an SSL connection to your client. When you connect to the server, part of the connection negotiation is passing the certificate to your client machine, the Browser will then check with the 'collection of trusted root certificates' in your computer and if it validates as authentic, your browser will accept a secure connection to the server.

How does your computer know which certificates to trust? It's usually via your browser, during install and updates.

For Microsoft Users: [https://technet.microsoft.com/en-us/library/cc754841\(v=ws.11\).aspx](https://technet.microsoft.com/en-us/library/cc754841(v=ws.11).aspx)

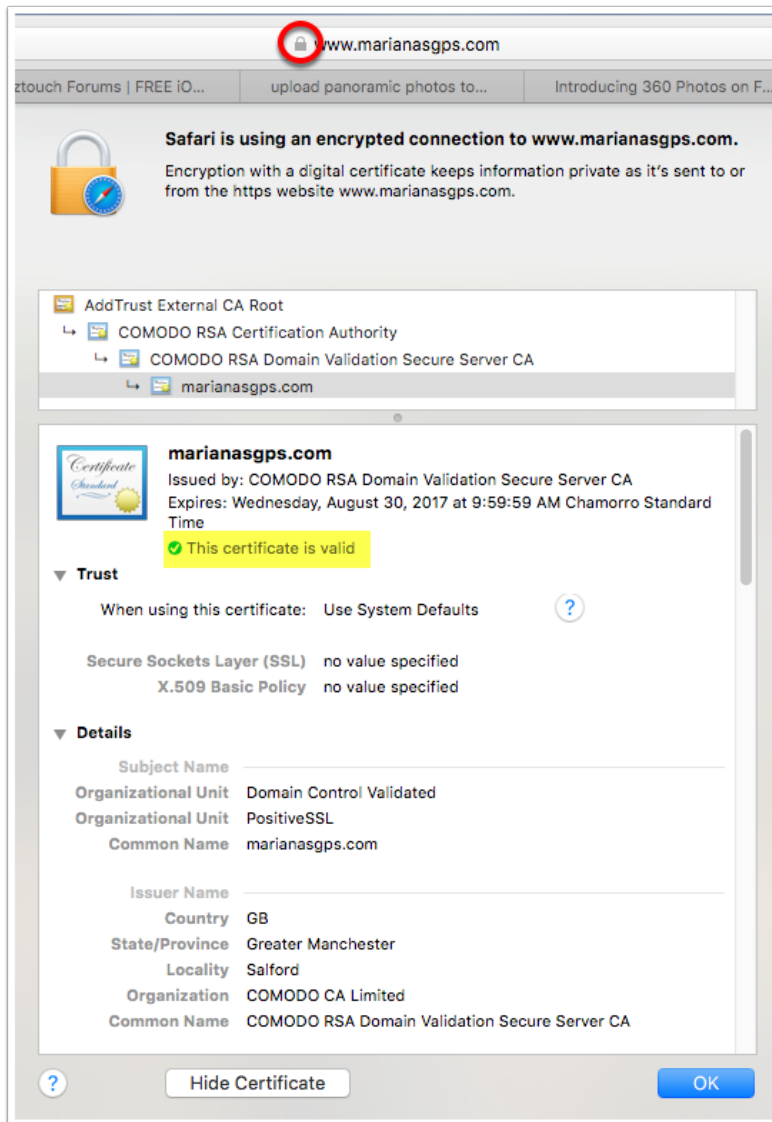
For Macintosh Users: <https://support.apple.com/en-us/HT202858>

People went through the same steps we are going to take, with much greater oversight and far more powerful machines, to create a file less than the size of a blank MS Word document. And they paid a lot to be verified as a trusted root organization. So there is a root. Usually that root will then grant privileges to other 'root' qualified companies... for a price. And it trickles down to you and your server. For a price. Per year, usually.

The 'root server key' holds the private key for every server that it has created a record for. When a connection to port 443 arrives at a webserver, it establishes a connection via SSL. This is all hardware/software. But the trust comes into play when your browser goes to create a secure connection, and asks for credentials from the server. The server supplies its 'public' certificate. The public certificate in some way jives with the private key that is contained in the root store certificate. If the certificates do not match, an alert will show bringing a discrepancy to your attention.



# Creating your own SSL Certificate





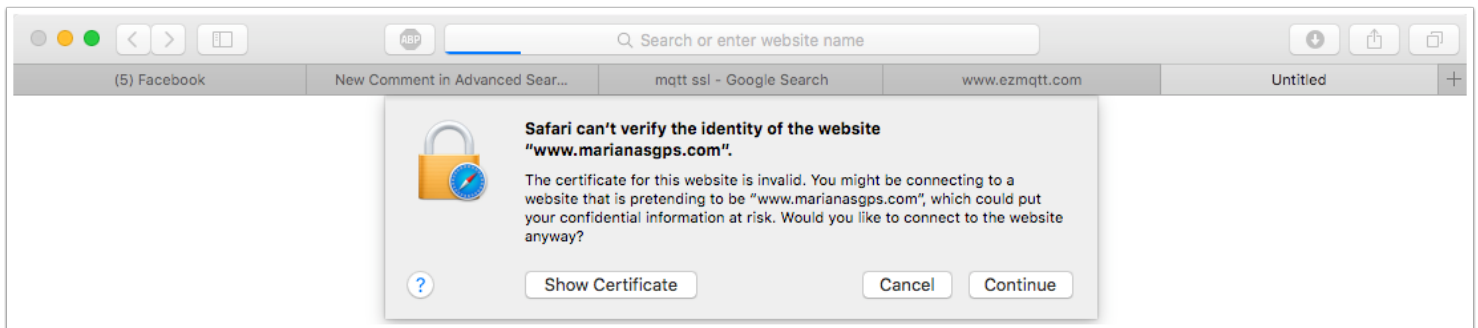
## When things aren't right

Things need to match up before they can be validated as a legitimate SSL connection. If things don't match, you get an alert.

Typically this means one of 3 things:

1. The Certificate expired and needs renewal
2. The Server URL on the certificate doesn't match the Server URL requested (name mismatch)
3. The Certificate is not installed in your Root Store

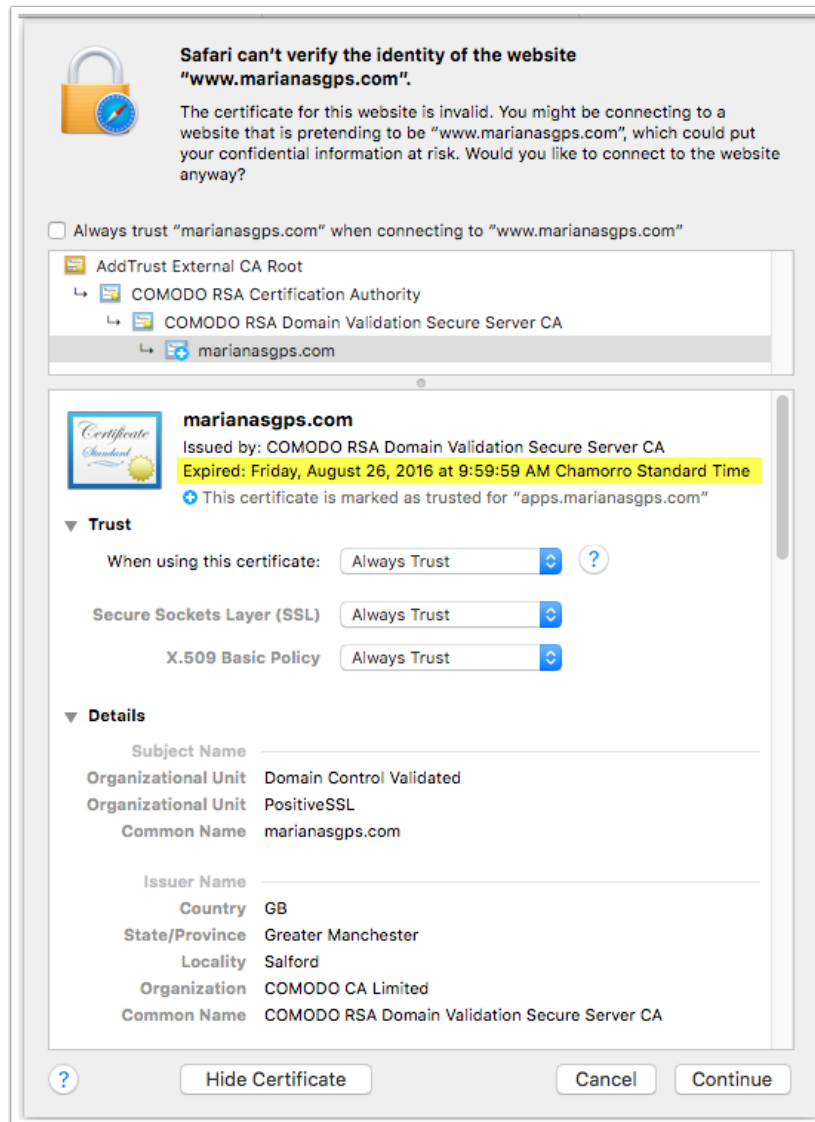
In these cases, it's best to examine the certificate before proceeding, or else just do not proceed.





## Expired Certificate, for Example

Oddly enough, when I thought about doing this tutorial, my SSL cert on my main server expired. How fortunate. When I went to view my site, it gave me a dialog. Of course, I was pretty sure I knew what happened, but I always check to be sure. And sure enough, my certificate expired.







## Name Mismatch, For Example.

My certificate was issued to 'www.marianasgps.com'. The way the certificate was created, means that "www.marianasgps.com" is the ONLY valid server address assigned to this certificate. This is what you'll see if you try to access 'https://apps.marianasgps.com'

**Safari can't verify the identity of the website "apps.marianasgps.com".**

The certificate for this website is invalid. You might be connecting to a website that is pretending to be "apps.marianasgps.com", which could put your confidential information at risk. Would you like to connect to the website anyway?

Always trust "marianasgps.com" when connecting to "apps.marianasgps.com"

AddTrust External CA Root

- COMODO RSA Certification Authority
  - COMODO RSA Domain Validation Secure Server CA
    - marianasgps.com**

**marianasgps.com**

Issued by: COMODO RSA Domain Validation Secure Server CA  
Expires: Wednesday, August 30, 2017 at 9:59:59 AM Chamorro Standard Time

**This certificate is not valid (host name mismatch)**

**Trust**

When using this certificate: Use System Defaults

Secure Sockets Layer (SSL) no value specified

X.509 Basic Policy no value specified

**Details**

Subject Name

Organizational Unit Domain Control Validated

Organizational Unit PositiveSSL

Common Name marianasgps.com

Issuer Name

Country GB

State/Province Greater Manchester

Locality Salford

Organization COMODO CA Limited

Common Name COMODO RSA Domain Validation Secure Server CA

Buttons: Hide Certificate, Cancel, Continue

**You can override at your own risk**



## Not in Root Store, For Example



## Obtaining Open SSL

You'll need Open SSL. There are various places to get it from, and Open SSL themselves do not 'advocate' any one vendor. Still, it's better to get something linked off their website.

<https://wiki.openssl.org/index.php/Binaries>

The safest way to get Open SSL is to download from [openssl.org](https://www.openssl.org) and compiling it yourself.

<https://www.openssl.org/source/>

The Windows version I use I obtained from here: <http://slproweb.com/products/Win32OpenSSL.html>

And regardless of which platform you're on, it's going to be a command line experience, so buckle up Ringo.

```
Administrator: Command Prompt - openssl
C:\SSL>openssl
OpenSSL> _
```



## What are we doing here?

We're doing a number of things.

- Creating a Private Key for the Server (This key keeps a record of all certificates that you create, and it must not be lost or compromised)
- Creating a Server Certificate Authority (This is the root certificate that establishes the server identity according to 'you'. People need to trust your certificates, ie trust 'you', before they can install these certificates on their machines. all certificates you create will be based off this root certificate.

When those two files are created, they are the most important files you can imagine, if you're doing this for more than fun. They, in essence, represent 'YOU', and the ability for you to safeguard that information. Now, genuinely, for folks like you and I, there isn't much at risk. For a Bank or an Insurance Company, a breach of security could cost millions of dollars.

Once that is done, your server has an identity, and the server can 'create' SSL certificates from requests. The Machine used to issue the SSL Certificates does not have to be the same server the service is on. In fact, you actually don't want it on the same machine. Again, in all those professional places, you have locked doors and sealed buildings and armed guards and all kinds of crazy stuff to ensure that the root credentials cannot be compromised.

This is the part where you want to take your 'credentials' and put them on a USB Stick, and place it in a fireproof safe, safe deposit box, or buried with the tomato plants in the garden. The thing is, keep it safe. Those are your 'root' credentials and similar to financial information, hackers can use these files to clone your server 'identity'. Bad Juju.

Now that your Certificate Authority is up and running, it's time to

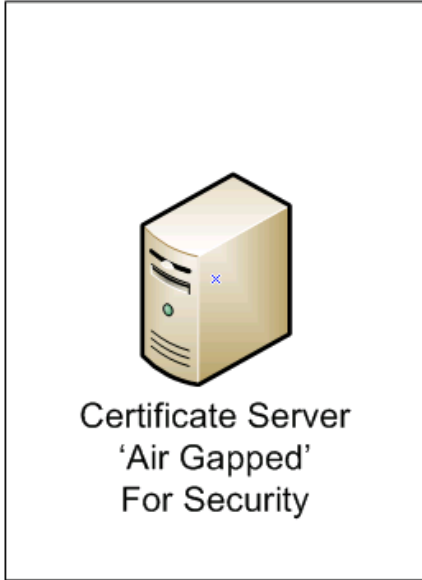
- Create a Certificate Request (You have to create a request with lots of information about the target server in the form of a CSR before you can grant one.)
- Create a Certificate from a Request (Creating the certificate for the requested entity)

For this example, I'm going to do 'everything' on the same machine. But again, 'best practices' require you to have all kinds of separation so that the credentials cannot be compromised. It all comes down to risk management. What are you willing to do? How far are you willing to go? But for now, for training purposes, let's do it on the same machine.



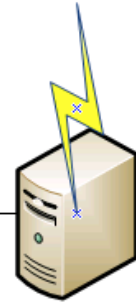
# Creating your own SSL Certificate

In a perfect (or perfectly expensive) world...



Create a 'CSR' (Certificate Signing Request) on a local workstation. This CSR has information regarding the Server to be used.

Copy the CSR onto a USB stick, hand carry it into the 'vault' and use the Certificate Server to generate a certificate from the CSR. Copy the certificate onto the USB stick. Copy it back to the workstation, and upload it to the Server.



Webserver, available  
on the internet



## SSL Configuration File (sslconf.txt)

I named it sslconf.txt, you can name it whatever you want. But for the moment, that is the filename we are going to refer to. Keep it straight.

The SSL Conf file is, of course, the basic configuration for your certificates. All of them. So make sure the information is correct.

1. The name of your website keyfile. unique. important.
2. Your ISO 2 Letter Country Designator
3. Your City Name
4. Your Company Name
5. Your Name
6. Your Email Address

save it into the same directory as your openssl.exe file



```
[ req ]
default_bits          = 2048
default_keyfile       = myWebsiteKey.pem 1
distinguished_name   = req_distinguished_name
attributes            = req_attributes
x509_extensions      = v3_ca

dirstring_type = nobmp

[ req_distinguished_name ]
countryName          = Country Name (2 letter code)
countryName_default  = US 2
countryName_min      = 2
countryName_max      = 2

localityName         = Locality Name (eg, city)
localityName_default = Tamuning 3

organizationalUnitName = Organizational Unit Name (eg, section)
organizationalUnitName_default = Certifying Authority

organizationName     = Organization (eg, who's asking)
organizationName_default = Your Company Name 4

commonName           = Common Name (eg, YOUR name)
commonName_default   = Your Name 5
commonName_max       = 64

emailAddress         = Email Address
emailAddress_default = your@emailaddress 6
emailAddress_max     = 40

[ req_attributes ]
challengePassword     = A challenge password
challengePassword_min = 4
challengePassword_max = 20

[ v3_ca ]

subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid:always,issuer:always
basicConstraints = CA:true
```



## Generating your server key

This will be your server identity. This is the important one, because from this all else is created. Protect this. Keep it safe. And most important, keep it.

- Open a Dos Prompt.
- Navigate to the OpenSSL directory.
- At the dos prompt type: **openssl genrsa -des3 -out yourserverkey.pem 2048**
- Type and retype a passphrase. Please please please remember it.
- Your Server Key file is created in the openssl directory. A record of all the certs you create for others will be kept in this file.

It's great for testing and all, but be sure to rename 'yourserverkey.pem' to something a bit more germane. (domain\_dot\_com\_certificate\_server\_key.pem)

File Name	Modified	Type	Size
libeay32.dll	1/28/2006 2:10 PM	Application extension	1,493 KB
libssl32.dll	1/28/2006 2:11 PM	Application extension	598 KB
openssl.exe	1/28/2006 2:14 PM	Application	1,535 KB
sslconf.txt	1/28/2006 2:07 PM	Text Document	2 KB
<b>yourserverkey.pem</b>	<b>8/30/2016 12:56 AM</b>	<b>PEM File</b>	<b>2 KB</b>

```
C:\SSL>openssl genrsa -des3 -out yourserverkey.pem 2048
Loading 'screen' into random state - done
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
Enter pass phrase for yourserverkey.pem:
Verifying - Enter pass phrase for yourserverkey.pem:
C:\SSL>_
```





## Generating your Server Root Certificate

This creates your Server Root Certificate. This is the 'authority' that allows you to create 'child' certificates for other machines. 'If' they trust you.

- Still got your dos prompt open? Good. If not, reopen one, get back to the openssl directory.
- At the dos prompt type: **openssl req -new -x509 -keyout yourserverkey.pem -out yourserver.crt -config sslconf.txt -days 365**
- type and retype the server key passphrase (so it can write it's first certificate record and private key, your root cert, into the file)
- Edit or Accept the values presented to you
- Your Server Root Certificate is now available in your openssl directory. Make a copy of it and your key, and hide them someplace safe.

The last value '365' as you may guess, is the number of days the certificate will be valid. How do you change it? You change it. I'm not sure what the maximum value allowed is, but I usually create my server certs for about 5 years.

The reason a lot of vendors allow only a 1 year certificate is usually policy based. It's all about trust, and risk management. They want to make sure that nothing with the server has changed over the last year. Oh, and making more money is also a driving force. It takes no more than a simple edit to change 365, to 1825 (5 years). But there you go. If it were easy, everyone would be doing it.



# Creating your own SSL Certificate

libeay32.dll	1/28/2006 2:10 PM	Application extension	1,493 KB
libssl32.dll	1/28/2006 2:11 PM	Application extension	598 KB
openssl.exe	1/28/2006 2:14 PM	Application	1,535 KB
sslconf.txt	8/30/2016 1:04 AM	Text Document	2 KB
yourserver.crt	8/30/2016 1:05 AM	Security Certificate	2 KB
yourserverkey.pem	8/30/2016 1:05 AM	PEM File	2 KB

```
C:\SSL>openssl req -new -x509 -keyout yourserverkey.pem -out yourserver.crt -config sslconf.txt -days 365
Loading 'screen' into random state - done
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to 'yourserverkey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:

You are about to be asked to enter information that will be included
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) [US]:
Locality Name (eg, city) [Amuning]:
Organizational Unit Name (eg, section) [Certifying Authority]:
Organization (eg, whos asking) [Your Company Name]:
Common Name (eg, YOUR name) [Your name]:
Email Address [your@emailaddress]:

C:\SSL>
```



## Generating a Server Certificate Request










This example will be using a webserver for the signing request. But it could be any kind of server; the only real requirement is to get the FQDN or IP Address correct.

- At the dos prompt, type: `openssl req -new -nodes -keyout mywebserver.key -out mywebserver.csr`
- Type the appropriate value for your country.
- Type the appropriate value for your State or Province
- Type the appropriate value for Your City
- Type the appropriate value for your Company or Organization
- Type the appropriate value for your Unit or Department Name
- Type the appropriate value for the server FQDN (`www.yourserver.com`)
- Type the appropriate value for the contact email address
- Create a new passphrase for this request. Different than your server key passphrase.
- Type the appropriate value for an optional company name, if appropriate.

The Certificate Signing Request, as well as the Server (webserver, not certificate server) key is created in the openssl directory.



# Creating your own SSL Certificate

	libeay32.dll	1/28/2006 2:10 PM	Application extension	1,493 KB
	libssl32.dll	1/28/2006 2:11 PM	Application extension	598 KB
	openssl.exe	1/28/2006 2:14 PM	Application	1,535 KB
	sslconf.txt	8/30/2016 1:04 AM	Text Document	2 KB
	yourserver.crt	8/30/2016 1:05 AM	Security Certificate	2 KB
	yourserverkey.pem	8/30/2016 1:05 AM	PEM File	2 KB
	.rnd	8/30/2016 1:21 AM	RND File	1 KB
	myserver.key	8/30/2016 1:22 AM	Registration Entries	2 KB
	server.csr	8/30/2016 1:22 AM	CSR File	2 KB

```
Administrator: Command Prompt
C:\SSL>openssl req -new -nodes -keyout myserver.key -out server.csr
Loading 'screen' into random state - done
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to 'myserver.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]:Guam
Locality Name (eg, city) []:Tamuning
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Marianas GPS
Organizational Unit Name (eg, section) []:Information Technology Dept
Common Name (e.g. server FQDN or YOUR name) []:test.marianasgps.com
Email Address []:info@marianasgps.com

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

C:\SSL>_
```



## Creating the Certificate from a CSR

At the dos prompt

```
openssl x509 -req -days 365 -in server.csr -CA yourserver.crt -CAkey certServer.pem -CAcreateserial -out server.crt
```

- server.csr is the CSR you just created.
- yourserver.crt is the Certificate Server Root Certificate
- yourserverkey.pem is the Certificate Server key pem file
- server.crt is the name for your new Web Server SSL Certificate.

If you want more than a year, now is the time to do it. Adjust the -days value accordingly.

After you've created your cert, you can take a look at the values. Ensure they are correct and reflect desired configuration. Especially with the FQDN.



# Creating your own SSL Certificate

The screenshot shows a Windows file explorer window with the following files:

File Name	Modified	Type	Size
libeay32.dll	1/28/2006 2:10 PM	Application extension	1,493 KB
libssl32.dll	1/28/2006 2:11 PM	Application extension	598 KB
openssl.exe	1/28/2006 2:14 PM	Application	1,535 KB
sslconf.txt	8/30/2016 1:04 AM	Text Document	2 KB
yourserver.crt	8/30/2016 1:05 AM	Security Certificate	2 KB
yourserverkey.pem	8/30/2016 1:05 AM	PEM File	2 KB
.rnd	8/30/2016 1:21 AM	RND File	1 KB
myserver.key	8/30/2016 1:22 AM	Registration Entries	2 KB
server.csr	8/30/2016 1:22 AM	CSR File	2 KB
server.crt	8/30/2016 1:40 AM	Security Certificate	2 KB
yourserver.srl	8/30/2016 1:40 AM	SRL File	1 KB

The Command Prompt window shows the following commands and output:

```
C:\SSL>openssl x509 -req -days 365 -in server.csr -CA yourserver.crt -CAkey yourserverkey.pem -CAcreateserial -out server.crt
Loading 'screen' into random state - done
Signature ok
subject=/C=US/ST=Guam/L=Tamuning/O=Marianas GPS/OU=
N=test.marianasgps.com/emailAddress=info@marianasgps.com
Getting CA Private Key
Enter pass phrase for yourserverkey.pem:
C:\SSL>
```

The 'Certificate' dialog box displays the following information:

- Issued to:** test.marianasgps.com
- Issued by:** Your Name
- Valid from:** 8/30/2016 to 8/30/2017

Buttons: Install Certificate..., Issuer Statement, OK



## An interesting trick...

At my home, I have a static IP address, so that I can run services from my home network and access them consistently from anywhere. In my DNS files, I've created a subdomain, "test" that I created an entry for. So when I type 'test.mydomain.com' it comes to my home router, and from there, it gets filtered and directed. Nothing special there.

But... lets say I have a test webserver on my internal network at IP Address '192.168.1.200'. If it's a windows server, edit your 'hosts' file (Windows/System32/Drivers/etc/) you will need admin privileges to edit this file. Add an entry with your 'internal' ip address, with the FQDN of the 'test' server (test.yourdomain.com, or whatever it really is). If you do this, the certificate you generate can use the domain name rather than an IP Address for the certificate hostname. It works like a treat.

If you're not running windows, the same concept should apply, just find your particular 'hosts' file and make the same changes.



## Where from here?

From here, you take your server certificate and install it into your server. There are many different servers, and all have already been well documented with respect to using an SSL certificate. I won't repeat it here.

But I hope this got you moving in the right direction, or at least cleared a few things up about how all this works, and why it does.

Feel free to ask questions in the BT Forums.

Cheers!

-- Smug



