Apple Pro Training Series

OS X Server 5.0 Essentials

Using and Supporting OS X Server on El Capitan

Arek Dreyer and Ben Greisler

Apple Certified

Lesson and media files available for download

Table of Contents

Configuring Accounts

Using Collaborative Services

Lesson 4

Configuring SSL Certificates

You can use OS X Server without doing any additional work to secure its services. However, you can use the Secure Sockets Layer (SSL) technology to prove your server's identity to client computers and devices and to encrypt communication between your server and client computers and devices. This lesson starts by describing the basics of SSL and then shows you how to configure SSL certificates for use with OS X Server.

GOALS

- Describe the basics of SSL certificates
- Create a certificate signing request
- Create a self-signed SSL certificate
- Import a certificate signed by a certificate authority
- Archive your certificate
- Renew your certificate
- Configure which certificate your OS X Server services use

Reference 4.1
Describe SSL Certificate Basics

You want the users who use your server's services to trust your server's identity and to be able to encrypt network traffic with your server.

The OS X solution is to use SSL, which is a system for transmitting data securely between hosts. You can configure your server to use an SSL certificate, which provides the ability to use the SSL system.

An SSL certificate (also referred to as simply a *certificate*) is a file that identifies the certificate holder. A certificate specifies the permitted use of the certificate and has an expiration date. Importantly, a certificate includes a public key infrastructure (PKI) public key.

PKI involves the use of public and private keys. Grossly simplified, a *key* is a cryptographic blob of data, and within PKI, public and private keys are created in a way that they are mathematically linked: Data encrypted with one key can be decrypted only by using the other key. If you can decrypt data with one key, it proves that the

data was encrypted with the other key. The public key is made publicly available, and the private key should be kept private. Fortunately, all of this encryption and decryption happens behind the scenes and is the basis for establishing secure communications.

Here are some definitions:

A *digital identity* (or more simply, an *identity*) is an electronic means of identifying an entity (such as a person or a server). An identity is the combination of a certificate (which includes the public key) and the corresponding private key. If you don't have your private key, you can't prove your identity. Similarly, if another entity has your private key, that other entity can claim your identity, so be sure to keep your private key private!

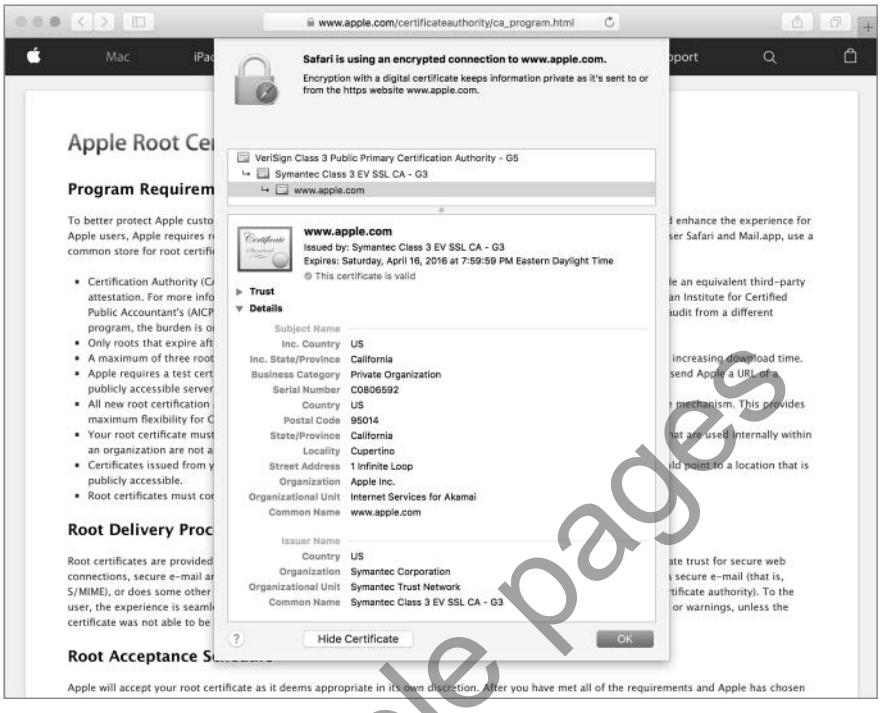
Again simplifying, a *digital signature* is a cryptographic scheme that uses PKI private and public keys to demonstrate that a given message (a digital file such as an SSL certificate) has not been changed since the signature was generated. If a message, which has been signed, changes or is otherwise tampered with, it will be clear that the signature no longer matches the underlying data. Therefore, you can use a digital signature on a certificate to prove its integrity.

A certificate must be either self-signed or signed by a *certification authority* (also known as a certificate authority or, more simply, a CA). In other words, you can sign your own certificate using your private key (remember that a certificate is a file that identifies the holder of the certificate and includes the public key), or you can have someone else, namely, a CA, use their private key to sign your certificate.

An intermediate CA is a CA whose certificate is signed by another CA. So, it's possible to have a hierarchical "chain" of certificates, where an intermediate CA, which in turn is signed by yet another CA, signs a certificate.

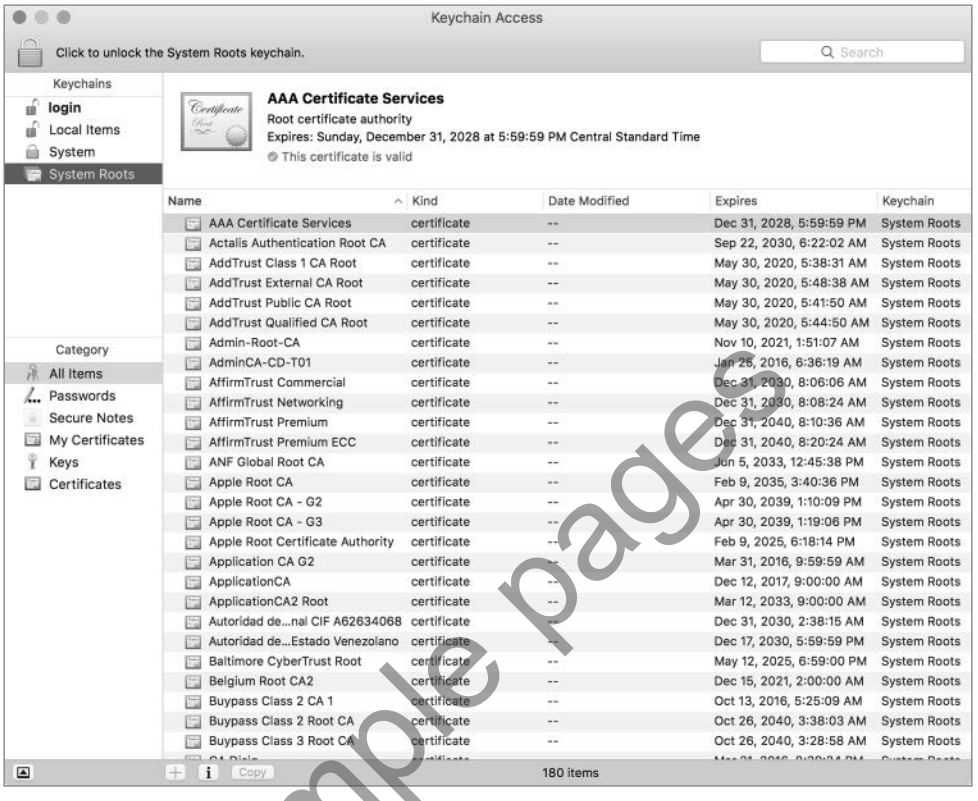
In the following figure, the certificate for www.apple.com is signed by an intermediate CA with the name of Symantec Class 3 EV SSL CA - G3, and that intermediate CA is signed by a CA with the name of Symantec Class 3 Public Primary Certification Authority - G5.

You can follow a chain of certificates, starting with a signed certificate, up to the intermediate CA and ending at the top of the chain. The certificate chain ends with a CA that signs its own certificate, which is called a root CA. You aren't required to have an intermediate CA—you could simply have a root CA sign your certificate—but in modern practice, an intermediate CA is often involved.



How do you know if you can trust a CA? After all, since a root CA has signed its own SSL certificate, this effectively means that the organization in control of a root CA simply asserts that you should trust that it is who it claims to be.

The answer is that trust has to start somewhere. In OS X and iOS, Apple includes a collection of root and intermediate CAs that Apple has determined are worthy of trust (see the Apple Root Certificate Program page on the Apple site for the acceptance process: www.apple.com/certificateauthority/ca_program.html). Out of the box, your Mac computers and iOS devices are configured to trust those CAs. By extension, your Mac computers and iOS devices also trust any certificate or intermediate CA whose certificate chain ends with one of these CAs. In OS X, these trusted CAs are stored in the System Roots keychain. (See Lesson 8, "Manage Keychain," in *Apple Pro Training Series: OS X Support Essentials 10.11* for more information about the various keychains in OS X.) You can use Keychain Access to view this collection of trusted root CAs. Open Keychain Access (in the Utilities folder). In the upper-left Keychains column, click System Roots. Note that in the following figure the bottom of the window states that there are at least 180 trusted CAs or intermediate CAs by default in El Capitan.



NOTE ► Some third-party software companies, such as Mozilla, do not use the System Roots keychain and have their own mechanism to store CAs that their software is configured to trust.

In Lesson 7, "Configuring Open Directory Services," you will learn that when you configure your server as an Open Directory master, the Server app automatically creates a new CA and a new intermediate CA and uses the intermediate CA to sign a new SSL certificate with your server's host name as the common name (the Common Name value is part of what identifies the certificate holder). If you haven't engaged a widely trusted CA to sign an SSL certificate for your server, you should use the SSL certificate signed by your Open Directory intermediate CA; in Lesson 9, "Configuring OS X Server to Provide Device Management," you will learn how to use the Trust Profile to configure your iOS devices and OS X computers to trust your Open Directory CA and, by extension, the intermediate CA and the new SSL certificate.

But what about computers and devices that are outside your control and that you cannot configure? When people use computers and devices that are not configured to trust your server's self-signed SSL certificate or your server's Open Directory CA or intermediate CA and they try to securely access services on your server, they will still see a message that the identity of your server cannot be verified.

One way to prove your identity is for your server to use an SSL certificate that's signed by a CA that most computers and devices are configured to trust or trust inherently.

Deciding What Kind of Certificate to Use

Before going through the process of getting a widely trusted CA to sign a certificate for you, consider the services you'll use with the certificate, as well as the computers and devices that will access those services.

If you use a self-signed certificate, there is no additional server configuration to install the certificate on your server, but you do need to configure each client to trust that self-signed certificate. For a Mac client, this involves not only distributing the certificate to the Mac and adding it to the System keychain but also configuring how the operating system will trust the certificate.

> **NOTE ►** If you use a self-signed certificate and are not able to configure all devices to trust that self-signed certificate, when users encounter a service that uses the self-signed certificate, a dialog informs them that the certificate may not be trustworthy and that to access services they must click Continue. This may undermine your efforts to train users not to automatically trust untrusted, expired, or otherwise invalid certificates.

If you use a certificate signed by a widely trusted CA, you need to generate a certificate signing request (CSR), submit the CSR to a CA, and then import the signed certificate.

Of course, you can use a mix of certificates for different services; if your Websites service responds to multiple host names, you'd want a certificate for each host name that you use for web services secured by SSL.

In all cases, you need to configure your server's services to use the appropriate certificates.

The next section shows you how to obtain a certificate that's signed by a widely trusted CA so that you can use it to prove the identity of your server and to encrypt communications between your server and the users of your server's services.

Reference 4.2
Configuring SSL Certificates

Your server has a default SSL certificate that's self-signed. That's a good start, but no other computers or devices will trust services that use that certificate without additional configuration. To get a CA to sign a certificate, start by using the Server app to create a certificate signing request. Specific steps to accomplish this objective follow in more detail, but generally they include the following:

- Generating a new CSR
- Submitting your CSR to a CA that is generally trusted
- Importing the signed certificate
- Configuring your server's services to use your newly signed certificate

The CA's process of using your CSR and signing your SSL certificate with its own private key includes verifying your identity (otherwise, why would anyone trust the CA if it signed certificates from unverified entities?) and optionally charging you money.

To finish the story, computers and devices can now use your server's services without getting a warning that your SSL certificate is not verified (as long as those computers and devices trust the CA you've chosen to sign your certificate). Additionally, your server and the users of its services can use your server's SSL certificate in the process of encrypting communications for services that use that SSL certificate.

Before you start creating new certificates, take a moment to inspect what you already have.

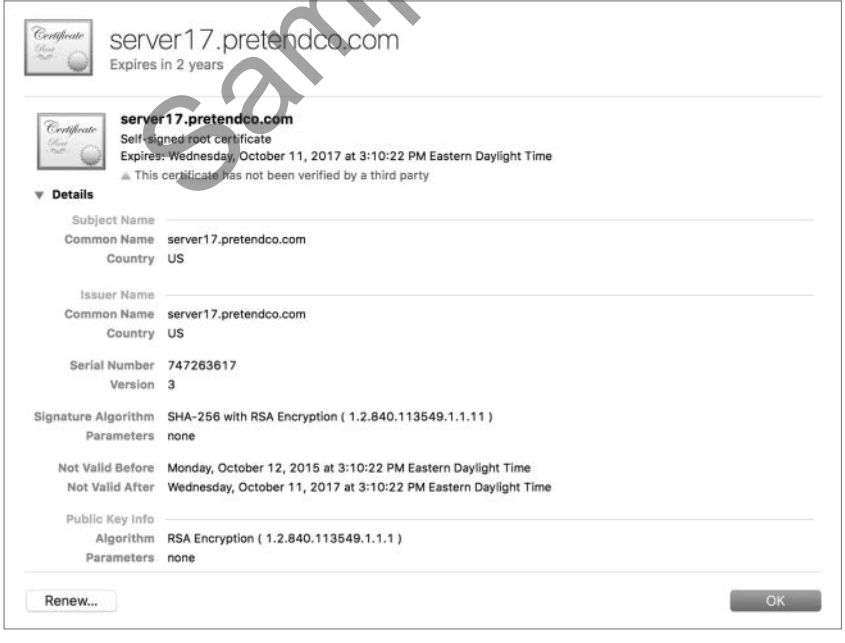
Viewing Your Server's Default Certificate

You can use the Server app to display certificates (if you're logged in at the server, you can also use the Keychain Access app). The standard behavior of Server app is to show all the certificates where earlier versions only showed some and you needed to pick the option to make all visible.

In the following figure, the certificate has the server's host name and expires in two years.

> **NOTE ►** When you use the Server app Change Host Name Assistant to change your server's host name, it automatically creates a new self-signed certificate for the new host name.

To get more details, double-click the certificate; alternatively, select it and click the Edit (pencil icon) button. You'll need to scroll to inspect all of the certificate's information.



Click OK to return to the Certificates pane.

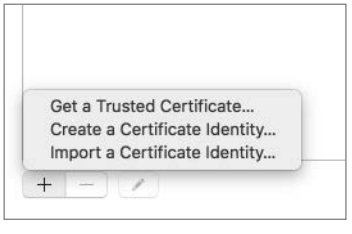
The following figure illustrates what you'd see after you configure your server as an Open Directory master or replica. At first glance, it looks like there is just one additional certificate, the code signing certificate, but the certificate with the server's host name is no longer a self-signed certificate but a certificate signed by your Open Directory CA; that certificate icon is blue, whereas the original self-signed certificate was bronze.

Explaining Options for Adding New Certificates

The existing self-signed certificates may not meet your needs. In the Server app Certificates pane, you have several options for adding a new certificate.

Click Add (+) to reveal three menu commands:

- Get a Trusted Certificate allows you to quickly generate a certificate signing request.
- Create a Certificate Identity is the command to choose to create a new self-signed certificate.
- Import a Certificate Identity allows you to import a signed certificate or a certificate and private key you've archived.



Obtaining a Trusted Certificate

You can choose to get a CA to sign a certificate for you so that users around the world can use your server's services without being notified that your server's identity is not verified.

At the bottom of the Certificates pane, click Add (+), and then choose Get a Trusted Certificate.

After that, you'll see the Get a Trusted Certificate assistant.

In the next pane, you can enter all the information necessary to establish an identity. A CA uses these details to verify your identity.

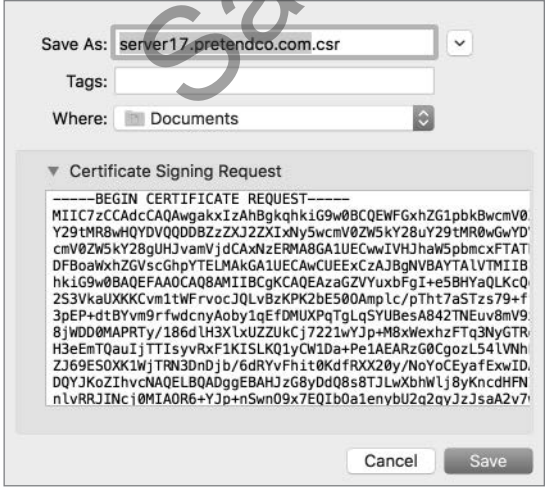
In the Host Name field, enter the host name you'll use for the services that will use this certificate. Use your organization's full legal name for the Company or Organization field, or if it's for personal use, just use your full name. The Department field is flexible; you can enter information such as your department name, but you should enter some value. To be fully compliant with standards, do not abbreviate your state or province. The following figure illustrates all the fields completed.

The next pane displays the text of your CSR, which you will submit to the CA of your choice. You can wait and access this text later, or you can select and copy this text, or click Save, now.

After you click Finish, the Server app displays the pending request.

If you didn't copy the text of your CSR earlier, you can access it again: Select the certificate marked Pending and click the Edit button (pencil icon), or just double-click the pending certificate item.

Your course of action depends on how your CA accepts CSRs. If your CA allows you to upload a text file, use the Save dialog to save the CSR as a text file. If your CA requires you to paste the text of the CA into a web form, click the disclosure triangle, and then copy the text of the CSR.



You need to choose an appropriate CA for your organization's needs (choosing a CA is outside the scope of this guide), send the CSR to the CA, and prove your identity to the CA. After some period of time, you will receive a signed certificate from the CA.

Importing a Signed Certificate

After you receive the signed certificate from the CA, you can import it with the Server app. If you are still at the list of certificates, double-click your pending certificate to reveal the field into which you can drag your signed certificate.

> **NOTE ►** If the CA provides you with the certificate in text form rather than in a separate file, you'll need to convert that text into a file. A quick way to do this is to copy the text, open TextEdit, press Command-N to create a new file, and choose Format > Make Plain Text (if that is an available command). Paste the text into the text file, and save it with a .cer extension.

Double-click the pending CSR, and drag the file containing a signed certificate, as well as any ancillary files provided by the CA, into the Certificate Files field (this is also where you could import a certificate and private key you've exported with Keychain Access). Once the certificate is in the Certificate Files field, its color will be blue, as long as the top of the certificate chain is a root CA your server trusts.

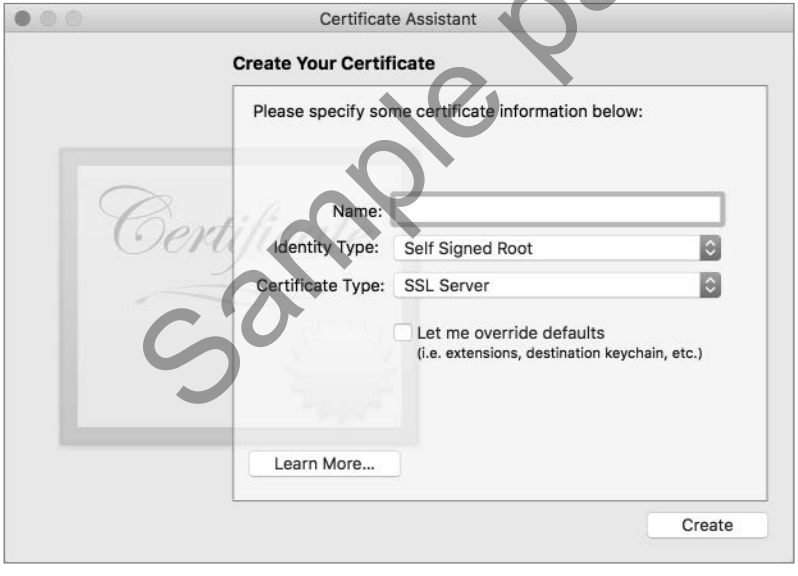
NOTE ► If you click Edit next to Certificate Request and then click Edit in the confirming dialog, a new public and private key pair and a new CSR will be generated, and you'll lose the original CSR.

Click OK to save your changes.

Generating a Self-Signed Certificate

In addition to generating a CSR, you can also use the Server app to generate a new self-signed certificate. This is useful if your server offers services at an alternative host name that corresponds to your server's Internet Protocol version 4 (IPv4) address or another IPv4 address your server is configured to use and if you have the ability to configure computers and iOS devices to trust the self-signed certificate.

In the Certificates pane, when you click Add (+) and choose Create a Certificate Identity, you see a blank Name field.

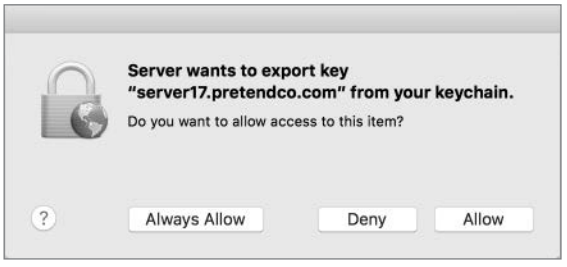


Enter the host name for the self-signed certificate, and then click Create.

NOTE ► You can select the "Let me override defaults" checkbox if you have more specific needs, but for most purposes, the defaults will suffice.

At the warning that you are about to create a self-signed certificate, click Continue.

At the Conclusion window, click Done. Finally, click either Always Allow or Allow to allow the Server app to copy the public and private key pair and the certificate from your login keychain to the System keychain and to /private/etc/certificates/.



You'll see the certificate in the Certificates field, with the bronze color that denotes a self-signed certificate.

Inspecting a Certificate

You can inspect your certificates with the Server app, as well as with the System keychain of your server computer (the System keychain contains items that are not user specific and that are available to all users of a system). The following figure shows a certificate that's been signed by a CA for test purposes. Note that the OS has not yet been configured to trust the CA that signed this certificate.

You can also use Keychain Access to inspect a certificate and its associated private key. Because the certificate and private key are stored in the System keychain on the server, you need to log in directly on your server (or use a screen-sharing method to control your server) to use Keychain Access to access the private key.

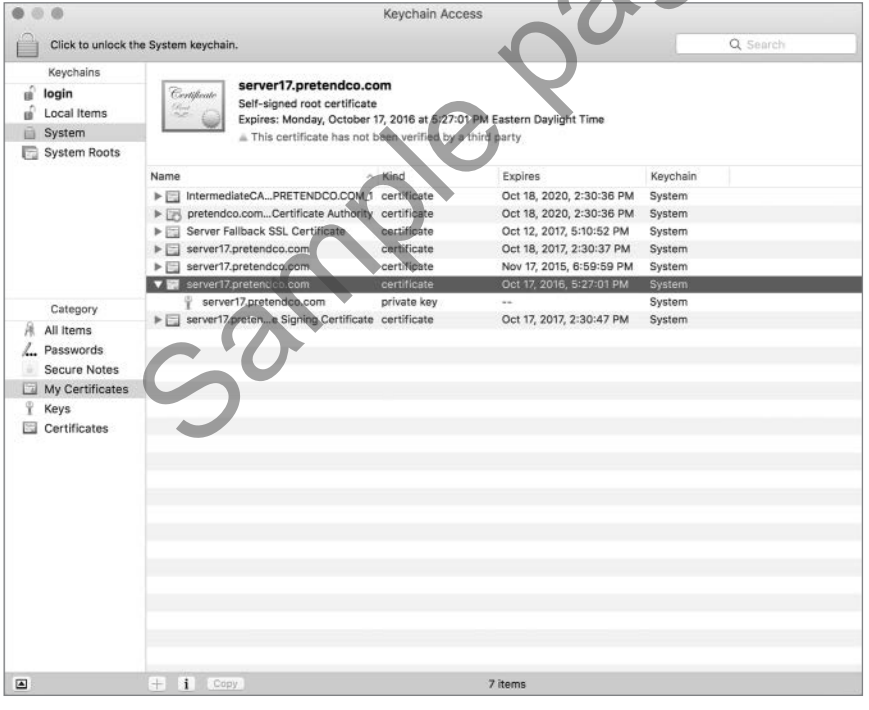
Keychain Access is in the /Applications/Utilities/ folder on your startup volume; you can use Spotlight or Launchpad to search for it (in Launchpad, it is in the folder named Other). Select the My Certificates category to filter the items that Keychain Access displays. If necessary, toggle the show/hide button in the lower-left corner of the Keychain

Access window until you can see all keychains. Select the System keychain to show items that are for the entire system, not just for the user who is currently logged in.

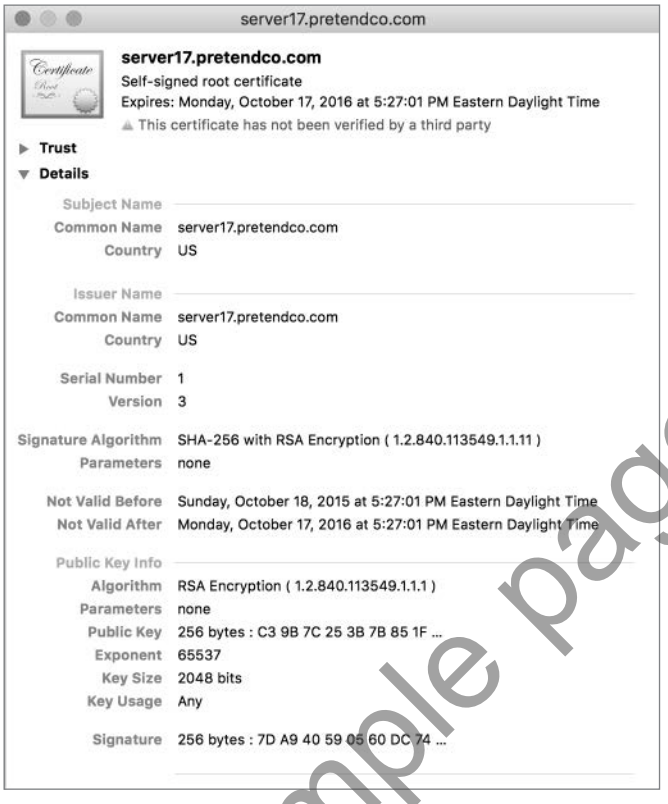
At least three items are listed (if you provided an Apple ID for push notifications, you will see more items):

- com.apple.servermgrd, which is used for remote administration with the Server app
- A certificate named Server Fallback SSL Certificate, which the Server app automatically uses if the default SSL certificate is removed
- An SSL certificate with the host name of your server

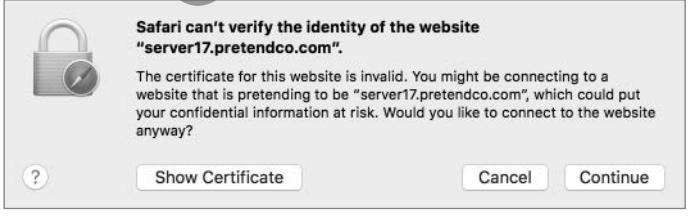
When you select a certificate that is not signed by a trusted CA, Keychain Access displays a warning icon, along with the text that explains the issue. In the following figure, the warning for the self-signed certificate is "This certificate has not been verified by a third party."



If you double-click your default self-signed SSL certificate to open it, you'll see a warning icon and the text "This certificate has not been verified by a third party."



If a service on your server uses this self-signed certificate, when users attempt to use services that use that SSL certificate, they may be warned that your SSL certificate is not trusted, as shown in the following figure.



Train your users that when they see an SSL warning, they should *not* continue using the service that uses the unverified SSL certificate.

Archiving Your Certificate

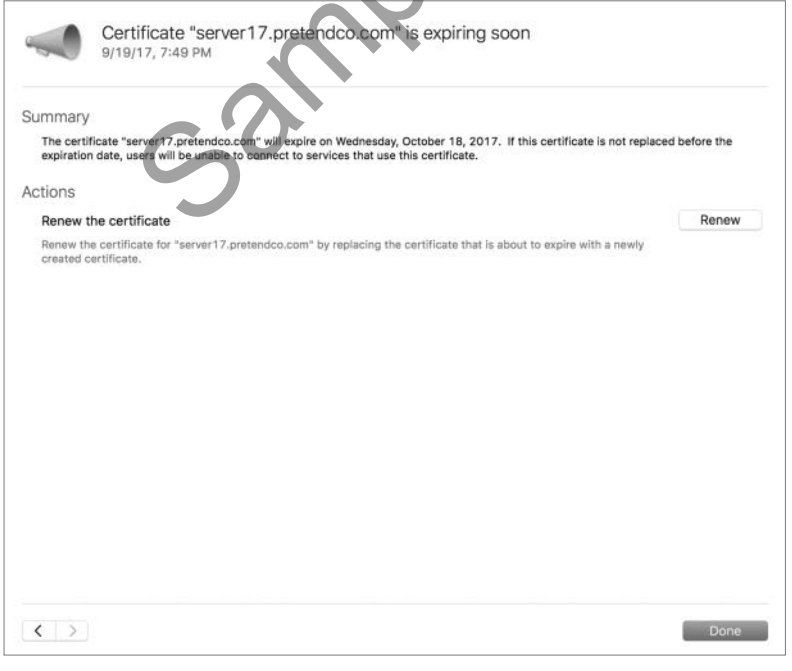
Whether you have a self-signed certificate or a certificate signed by a CA, you should take steps to archive your certificate and its private key. You may need to reinstall your server in the future, or an administrator might accidentally remove your certificate and its private key; if you have an archive of your certificate and private key, you can easily use the Server app to re-import your certificate and its private key.

You use the Keychain Access app to export your certificate and private key. Keychain Access prompts you to specify a password to protect your private key; make sure that you use a strong password.

You use the Server app to import the certificate and private key. You need to provide the password that was entered when the certificate was exported in the first place; otherwise, you will not be able to import.

Renewing Your Certificate

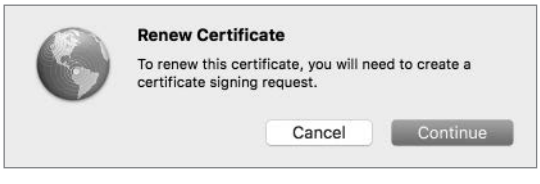
SSL certificates do not last forever. Luckily, renewing SSL certificates is simple. The Server app issues an alert when an SSL certificate expiration date approaches. To renew a self-signed SSL certificate, simply click Renew when viewing the certificate in the Certificates pane or when viewing the alert.



Once you click Renew, the Server app takes care of renewing the certificate, and the alert displays that the issue has been resolved.

NOTE ► Do not click Renew for an Open Directory CA because this causes changes to the CA properties, and your Open Directory intermediate CA will no longer be signed by a trusted authority.

If you have a certificate signed by a widely trusted CA, when you click Renew, you will see the message that you need to generate a new CSR. See the earlier section "Obtaining a Trusted Certificate" for more details.

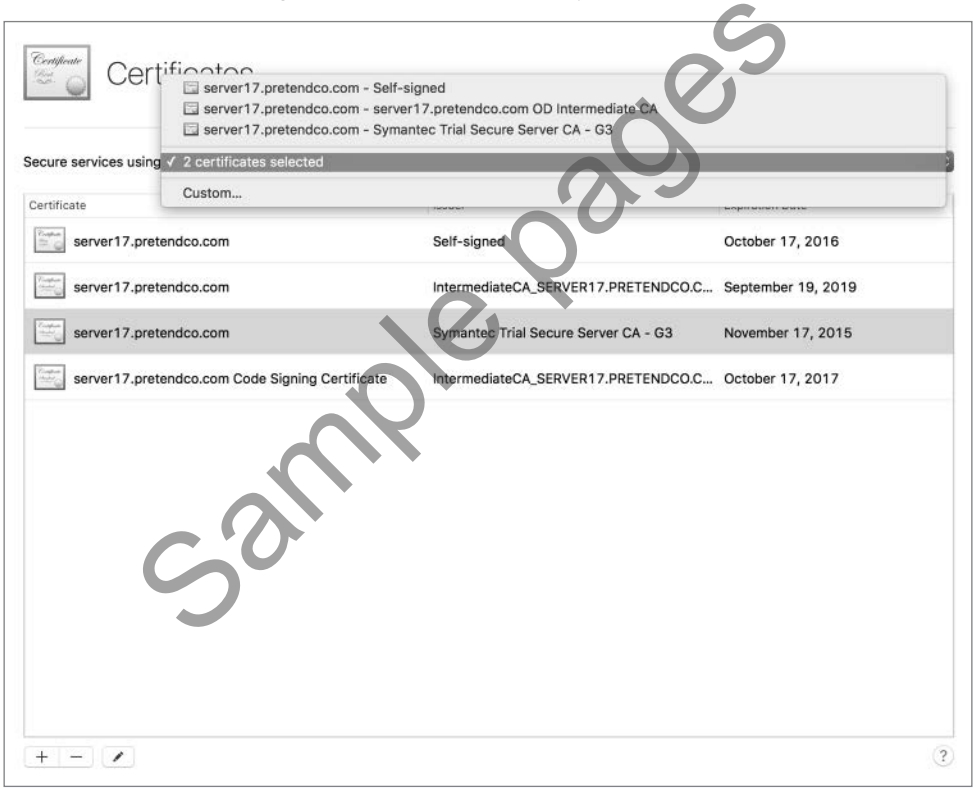


Configuring OS X Server Services to Use a Certificate

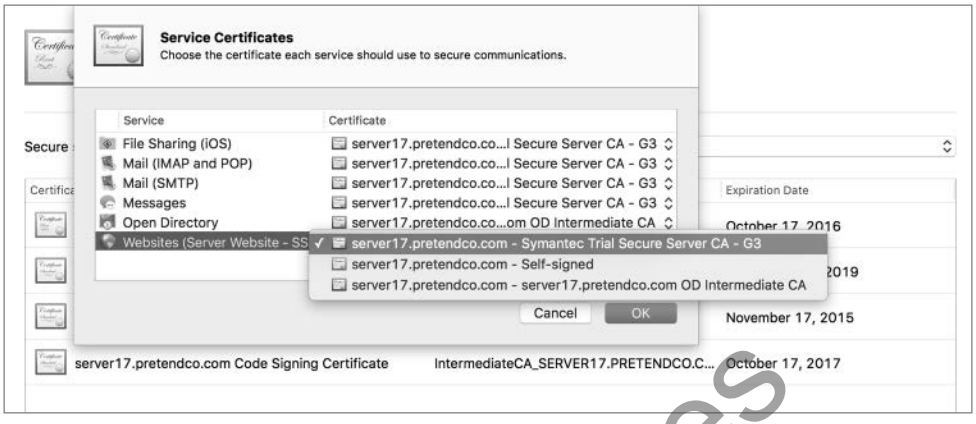
Once you have taken steps to obtain a signed certificate or create a new self-signed certificate or have configured your server as an Open Directory server, you should use the Server app to configure services to use that certificate. You start in the Certificates pane of the Server app.

With the pop-up menu, you can do either of the following:

- Choose one certificate to specify that all services use that certificate.
- Choose Custom to configure each service separately to use or not use a certificate.



The following figure shows an example of choosing Custom and then editing the value for the default secure site of the Websites service. Note that there are some extra certificates in the figure. This illustrates that you can configure your server to respond to requests at multiple host names, create a certificate for each host name, and configure each secure site to use the appropriate certificate.



You can use the Server app to configure the following OS X Server services to use SSL:

- File Sharing for iOS
- Mail (IMAP and POP)
- Mail (SMTP)
- Messages
- Open Directory (appears only after starting Open Directory services)
- Websites

You will see in Lesson 19, "Hosting Websites" that you can granularly specify an SSL certificate for each website you host, and you can use the Profile Manager pane to specify the SSL certificate to use for the Profile Manager service to sign configuration profiles.

A few other services use SSL but do not appear in the Server app:

- com.apple.servermgrd (for remote administration with the Server app)
- VPN
- Xcode
- Calendar and Contacts

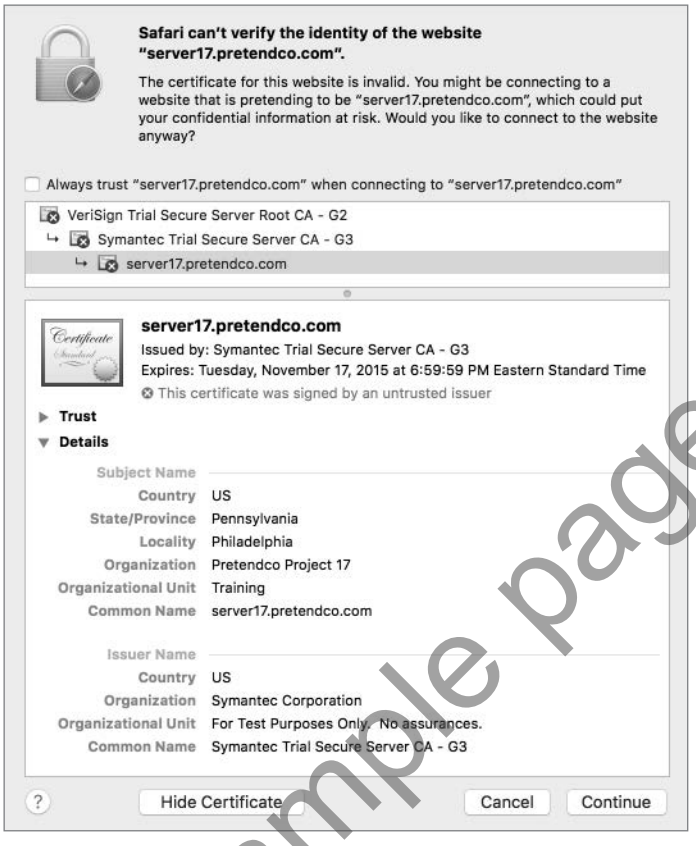
Following the Certificate Chain

When choosing a CA to use, make sure that it's a root CA that most computers and devices are configured to trust. Having a CA sign your certificate isn't useful if not many computers or devices will trust that certificate. As an example, the following figure shows how an SSL certificate signed by a trial CA appears in Keychain Access.

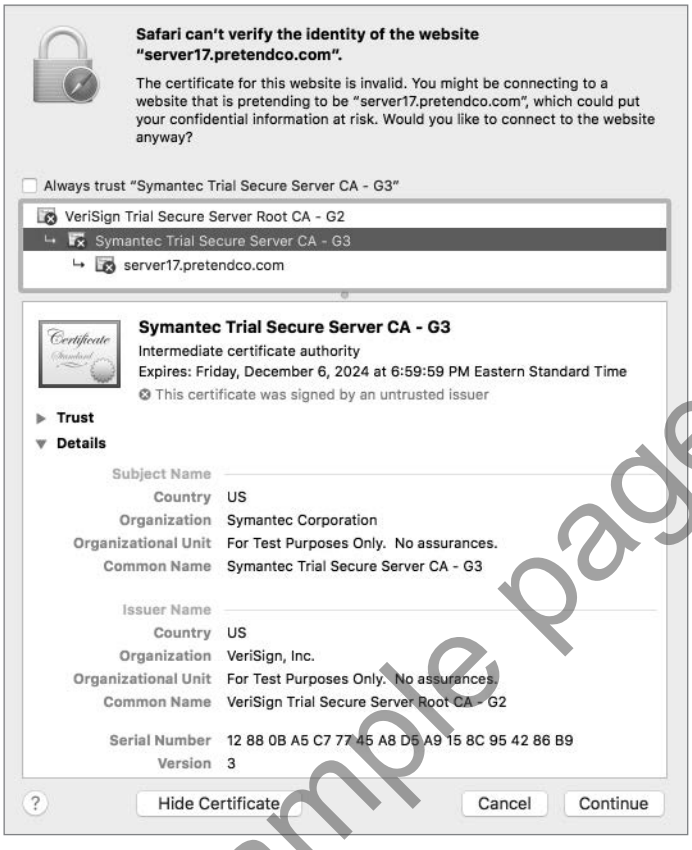
You can see that the "Issued by" field near the top of the window shows Symantec Trial Secure Server CA – G3. Note the red X icon and the text "This certificate was signed by an untrusted issuer." This is a CA that is by default not trusted by computers and devices, so even if you used this signed certificate for OS X Server services, the people who access your services would experience trouble. In some cases, the service might silently fail, or the user may be alerted that the identity of the service cannot be verified. The following figure illustrates that on a client Mac Safari notifies the user that Safari can't verify the identity of the website.

If you click Show Certificate, Safari displays the certificate chain. The following figure shows what you see when you select the server's certificate at the bottom of the certificate chain: that the certificate was signed by an untrusted issuer.

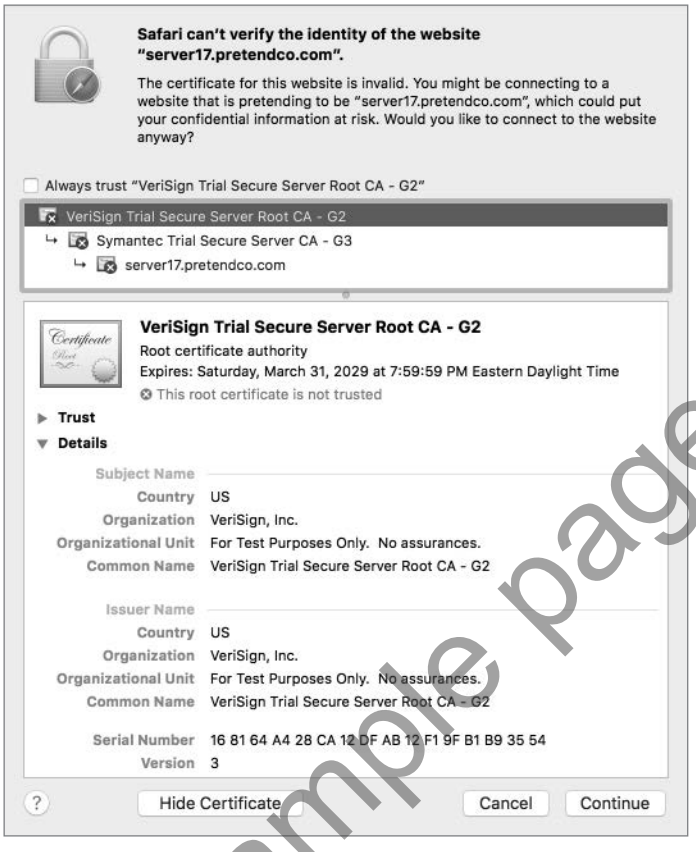
The following figure illustrates that if you click the Details disclosure triangle, you'll see information about the identity of the certificate holder, as well as information about the issuer (the entity that signed the certificate). In this case, the issuer's common name is Symantec Trial Secure Server CA – G3.



When you select the certificate in the middle of the certificate chain, you see that this is an intermediate CA; the window states "Intermediate certificate authority," and the Issuer Name information shows you that the common name of the issuer (or signer) is Symantec Trial Secure Server Root CA – G2.



Finally, when you select the certificate at the top of the certificate chain, you see that this is a root CA; the window states "This root certificate is not trusted." This root CA is not in this computer's System Root keychain, so Safari doesn't trust the intermediate CA, and it doesn't trust the server17.pretendco.com certificate either.



Since that example root CA is for trial use only, you should not configure your Mac to always trust it outside of a learning or testing environment.

Configuring Trust

You can configure your Mac to always trust a certificate for the currently logged-in user. Returning to the previous example of your server using its self-signed SSL certificate for a website, you can click Show Certificate and then select the "Always trust..." option.