OpenNETCF Consulting

HOL P111 – Securing a Padarn Site with SSL

Table of Contents

HOL Requirements

The following items are required to run this HOL:

- A Microsoft Windows Server 2003 or Server 2008 PC With the following services installed:
  - Internet Information Services (IIS)
  - Microsoft Certificate Authority
- Microsoft Visual Studio 2008 Professional (or higher)
- A Web Browser
- A Padarn reference system or developer kit

While Padarn will run on almost any hardware that supports the Microsoft .NET Compact Framework 2.0 or higher, this lab assumes that you have one of the Padarn reference hardware platforms with an OpenNETCF-validated Windows CE image running on it. If you are using an alternate hardware or software configuration, the steps outlined in this Hands-on Lab may not be accurate for your environment.

Summary

In this lab, you will learn the fundamentals of Secure Sockets Layer (SSL) including why it is an improvement over both Basic Authentication and Digest Authentication, details as to how Padarn implements SSL, and finally run through how Padarn responds to SSL requests and communicates with a client over HTTPS.

Lab Objective

Upon completion of this lab, you will be familiar with exchanges that place between a Padarn server and a requesting client when SSL is utilized.

In this HOL, you will perform the following exercises:

- ✓ Understand the terminology behind SSL exchanges
- ✓ Create a self-signed certificate to be used on your Padarn server for testing purposs
- ✓ Learn how to modify the ocfhttpd.exe.config file to enable SSL
- ✓ Test to ensure SSL is being used through a live request

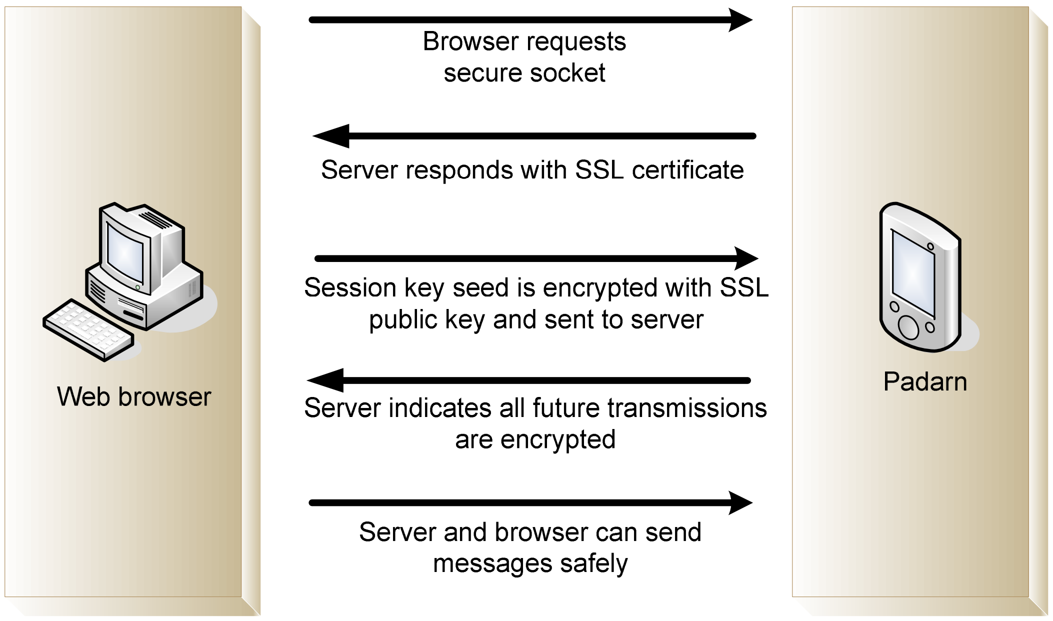
Background

In HOL P110 (Securing a Padarn Site with Digest Authentication), you learned how to enable Digest Access Authentication. This method is easy to configure and allows for somewhat secured transmission of data between a server and client; however, it does nothing to prevent a middle-man attack by which a third party can inject itself. For example, a website could pretend to be Bank of My Town's website, which costs several web service APIs for accessing balance information. Merely by using the same encryption routine as the client, it can intercept requests such as bank number and home address.

HTTP traffic performed over SSL uses the HTTPS (Hypertext Transfer Protocol over Secure Socket Layer) protocol. HTTPS provides assurance that the resource being accessed by the client can be verified as authentic. A Certificate Authority (CA) such as WebTrust and VeriSign can issue server administrators certificates that link to one of the many root certificates installed on nearly all Internet-connected computers (Windows will install about 50 certificates on a client OS installation). The benefit of this is that the client need not download any additional certificate files and can immediately uniquely identify the requested resource host.

SSL is the means by which all traffic transferred over HTTPS is encrypted. Padarn v1.1 supports SSL v3.1 (Transport Layer Security v1.0). Although there is a tremendous volume of documentation available about the SSL communication process, this description is intended to provide brief motivation for why it might be worthwhile to use Padarn's SSL support. When a client attempts to communicate with a Padarn web server over an SSL-protected HTTPS connection, a handshake takes place that gets the two devices "in sync" with respect to encryption method, keys to be used, and some random data to aide in the encryption process. In more detail:

1. The client sends a client "hello" message that lists the cryptographic capabilities of the client (sorted in client preference order), such as the version of SSL, the cipher suites supported by the client, and the data compression methods supported by the client. The message also contains a 28-byte random number.
2. Padarn returns its own "hello" message with random data to be used for encryption as well as other secure sockets layer information (including its SSL certificate – see further down for how to create such a certificate – with a long string of characters called a public key) that the client will need in step 4. It chooses from the strongest cipher suites supported by the client as indicated in the client's "hello" message. This message (after the SSL certificate public key) is sent, the server says "Hello Done".
3. The client checks the information it receives from the server and determines whether the SSL certificate is legitimate for the point in time (certificates have expiration dates) and resource URI (usually established by IP address). By leveraging an SSL certificate, a middle-man attack is ruled out, since a third party could steal your data even with encrypted packets being sent if that rogue server knows enough information about the encryption type and snoops for more clues as to the way data is being encrypted. If the server cannot be authenticated, the user is warned of the problem and informed that an encrypted and authenticated connection cannot be established. Most browsers like Firefox and IE allow this message to be ignored, though the various alerts are flashed to the user to remind him of the risks. Sometimes, client authentication is required by the server, but Padarn v1.1 does not support this feature.
4. Padarn sends a server "hello done" message and waits for a client response.
5. Upon receipt of the server "hello done" message, the client verifies the validity of the server's digital certificate and checks that the server's "hello" parameters are acceptable.

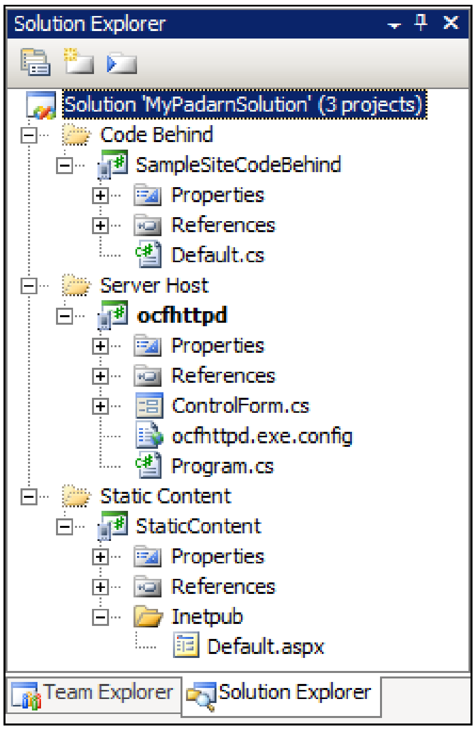

Web browser
Browser requests secure socket
Server responds with SSL certificate
Session key seed is encrypted with SSL public key and sent to server
Server indicates all future transmissions are encrypted
Server and browser can send messages safely
Padarn

At this point, the handshake process has outputted a key that will be used for all subsequent exchanges during this session. It's important to realize that the key is not just generic/random salt – it's based upon agreed upon values such as identity and SSL version. In more detail:

6. The client forms a pre-master secret that will be used to encrypt the remainder of the exchanges. This random key is used to encrypt with the agreed upon encryption method combined with the Padarn server SSL certificate public key string and sends this data back to the server. This is called the "client key exchange" message.
7. Using this pre-master secret string, both Padarn and the client create a new master secret string and use it to create session keys that are used for all subsequent communication – both during encryption and decryption - for the rest of the session. By using this master secret, both parties can make sure data wasn't altered in transmission. This master secret is considered symmetric since it is used in the same manner by both Padarn and the client.
8. The client now tells Padarn (in an encrypted message) that it is ready to proceed using the master secret
9. Padarn now tells the client that it is ready to begin using the master secret
10. If at any point the data being exchanged does not appear to be encrypted properly using the master secret, the handshake restarts from step 1.

Exercise 1: Configuring Padarn to use SSL Authentication

Because HOL 110 has instructions for creating a basic web page through a sample Padarn web server, it is recommended that you review that guide before proceeding. It is assumed that the various files in the included example solution are understood by the user. Here is what the solution looks like:



```
Solution Explorer
Solution 'MyPadarnSolution' (3 projects)
  Code Behind
    SampleSiteCodeBehind
      Properties
      References
      Default.cs
  Server Host
    ocfhttpd
      Properties
      References
      ControlForm.cs
      ocfhttpd.exe.config
      Program.cs
  Static Content
    StaticContent
      Properties
      References
      Inetpub
        Default.aspx
```

All security settings pertaining to a given Padarn instance exist within ocfhttpd.exe.config. The file exists within the ocfhttpd project already. There are two sections where we will focus our attention.

Verify the Padarn web server will use the newly-created Default.aspx as our default page. These settings exist within the `WebServer` section:

```
<WebServer
  LocalIP="0.0.0.0"
  DefaultPort="443"
  MaxConnections="20"
  DocumentRoot="\Windows\Inetpub\"
  Logging="true"
  LogFolder="\Temp\Logs"
  LogExtensions="aspx;html;htm;zip"
  BrowserDefinitions="\Windows\Inetpub\config\browsers"
  UseSsl="true"
  CertificateName="\Windows\certificate\server.pfx"
  CertificatePassword="4JY!jr"
>
```

First, be sure to change the `DefaultPort` to 443. While other values are supported, this is the standard port number for SSL transactions. Second, you'll notice that the `UseSsl` element is set to true. We will leave the `CertificateName` element as the default value. It's important to always use challenging passwords regardless of the level of risk you presume – old habits are hard to break! Change the `CertificatePassword` element to "4JY!jr". Should you generate a PFX-format certificate with a different password, you'll need to alter this value. Next, let's take a look at the Authentication mode section.

```
<Authentication Mode="Digest" Enabled="false" Realm="Padarn Tester Site">
  <Users>
    <User Name="adminuser" Password="adminpass" />
  </Users>
</Authentication>
```

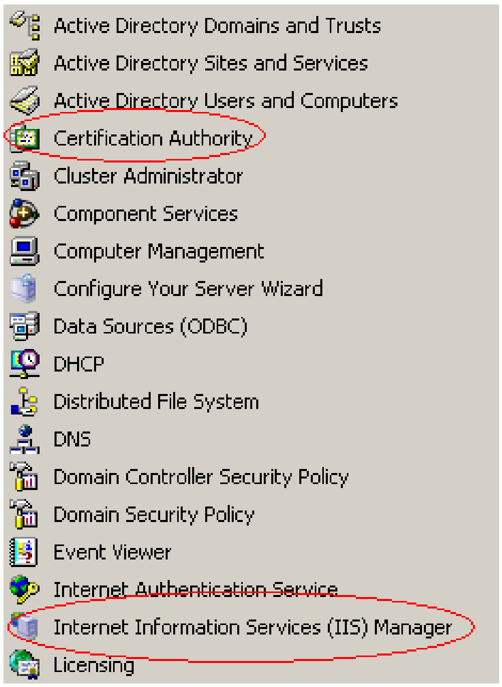
For simplicity sake, we will not be using an additional form of authentication; however, it is clearly a best practice to secure any website deemed worthy of encryption with an entry user name and password! Padarn supports both Basic and Digest Access Authentication over SSL.

Save the contents of the configuration file and be sure to deploy the entire solution to the device (the existing project properties should allow this to happen without further Visual Studio project setting changes). Select **Build -> Deploy Solution**.

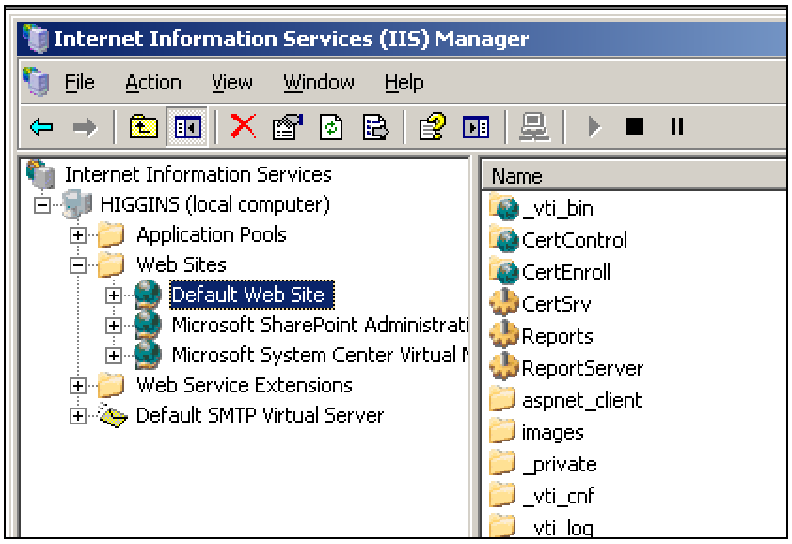
Exercise 2: Creating a test certificate

This exercise will step you through the process of generating a certificate (to be immediately installed into the cert store of your server) and then the appropriate exportable certificate file (to be dropped into the Padarn web solution). Because there are several third parties that are generally involved in issuing a valid server authentication certificate that will be accepted by all clients, it is beyond the scope of this article to illustrate a client experience with no warning messages. To get started creating a certificate within the store of your server:

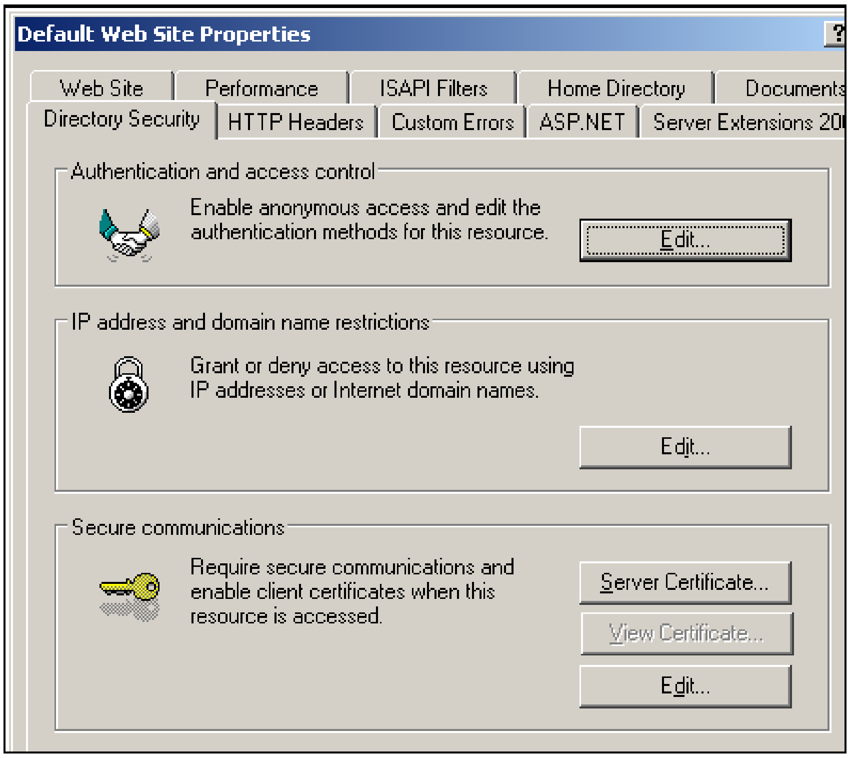
1. Open your Administrative Tools menu on your server and verify the Certification Authority and Internet Information Services (IIS Manager) are installed.


Active Directory Domains and Trusts
Active Directory Sites and Services
Active Directory Users and Computers
Certification Authority
Cluster Administrator
Component Services
Computer Management
Configure Your Server Wizard
Data Sources (ODBC)
DHCP
Distributed File System
DNS
Domain Controller Security Policy
Domain Security Policy
Event Viewer
Internet Authentication Service
Internet Information Services (IIS) Manager
Licensing

2. Open IIS Manager


Internet Information Services (IIS) Manager
File Action View Window Help
Internet Information Services
HIGGINS (local computer)
Application Pools
Web Sites
Default Web Site
Microsoft SharePoint Administrati
Microsoft System Center Virtual M
Web Service Extensions
Default SMTP Virtual Server
Name
_vti_bin
CertControl
CertEnroll
CertSrv
Reports
ReportServer
aspnet_client
images
_private
_vti_cnf
_vti_log

3. Right click Default Web Site (the website configured by IIS installation) with default content available

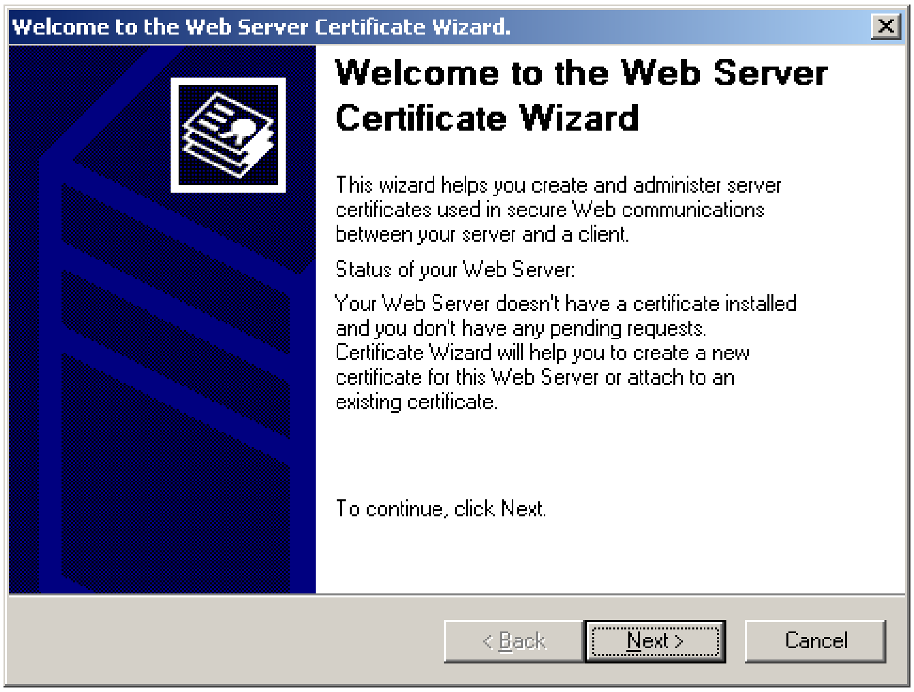


Enable anonymous access and edit the authentication methods for this resource.

Grant or deny access to this resource using IP addresses or Internet domain names.

Require secure communications and enable client certificates when this resource is accessed.

4. Press **Server Certificate**


Welcome to the Web Server Certificate Wizard.

Welcome to the Web Server Certificate Wizard

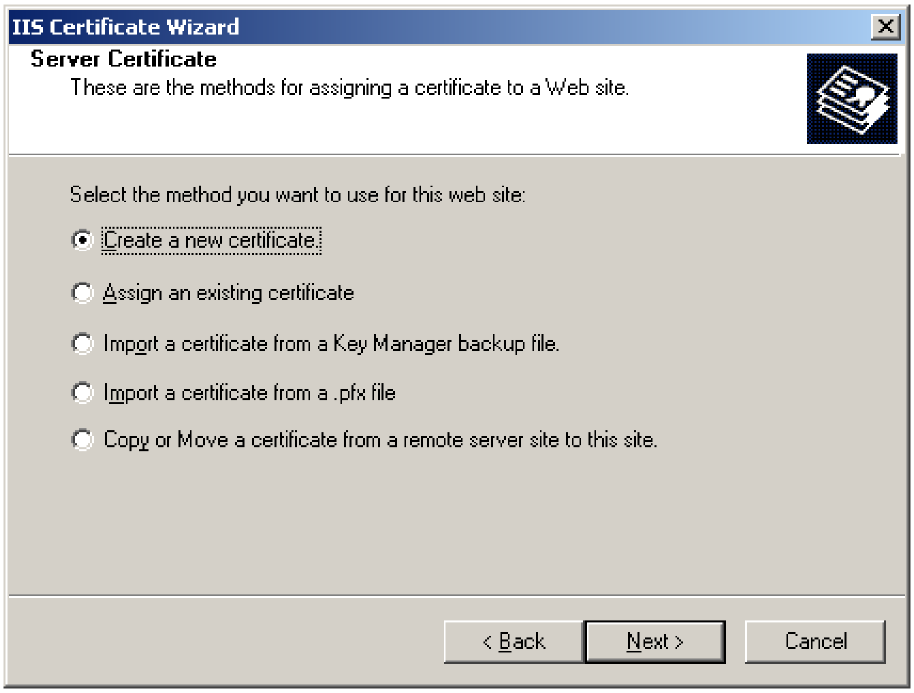
This wizard helps you create and administer server certificates used in secure Web communications between your server and a client.

Status of your Web Server:

Your Web Server doesn't have a certificate installed and you don't have any pending requests. Certificate Wizard will help you to create a new certificate for this Web Server or attach to an existing certificate.

To continue, click Next.

5. Press Next


IIS Certificate Wizard

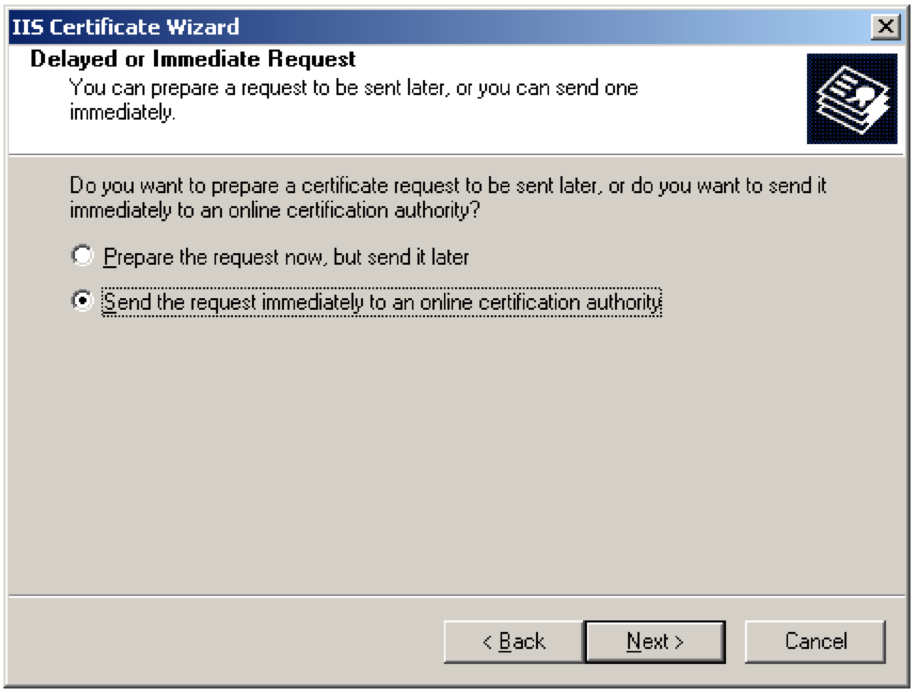
Server Certificate

These are the methods for assigning a certificate to a Web site.

Select the method you want to use for this web site:

- [✓ Create a new certificate.]
- Assign an existing certificate
- Import a certificate from a Key Manager backup file.
- Import a certificate from a .pfx file
- Copy or Move a certificate from a remote server site to this site.

6. Select “Create a new certificate” and select **Next**


IIS Certificate Wizard

Delayed or Immediate Request

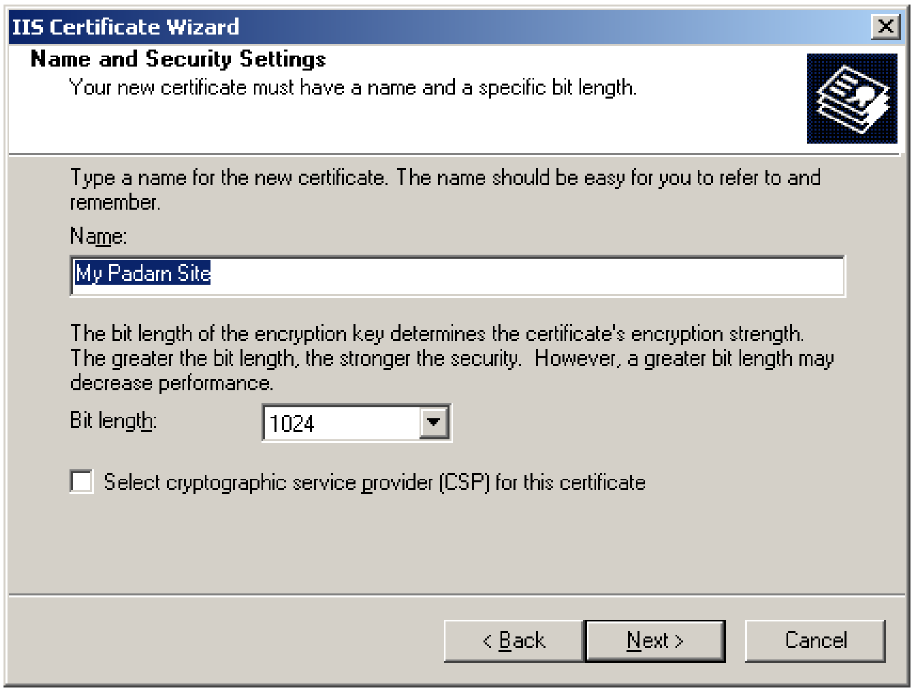
You can prepare a request to be sent later, or you can send one immediately.

Do you want to prepare a certificate request to be sent later, or do you want to send it immediately to an online certification authority?

○ Prepare the request now, but send it later

● Send the request immediately to an online certification authority

7. On the Delayed/Immediate Request dialog, select either choice. Sending a request to a real third party CA is beyond the scope of this lab.


IIS Certificate Wizard

Name and Security Settings

Your new certificate must have a name and a specific bit length.

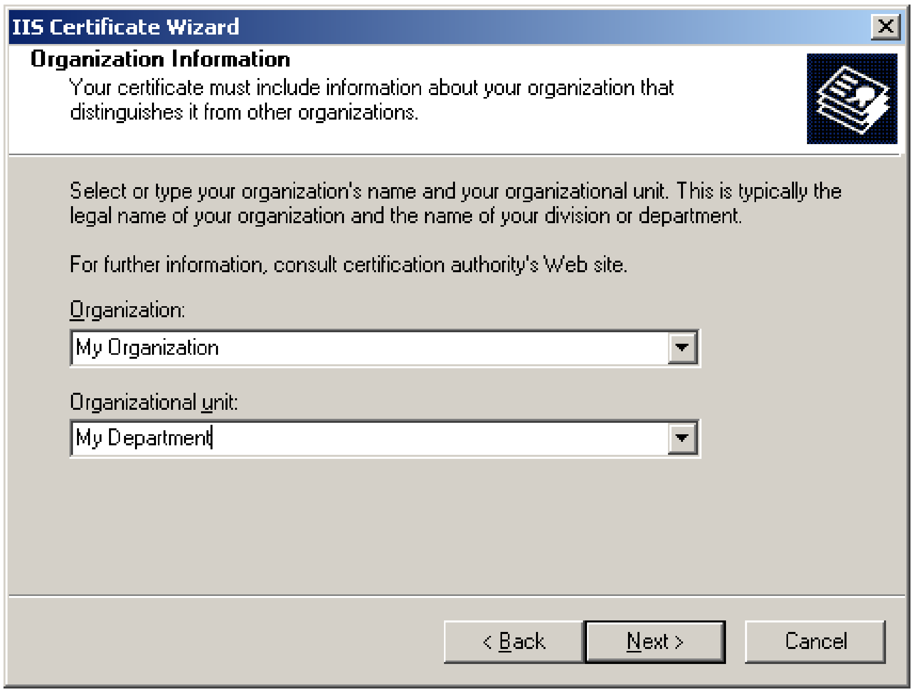
Type a name for the new certificate. The name should be easy for you to refer to and remember.

Name: My Padarn Site

The bit length of the encryption key determines the certificate's encryption strength. The greater the bit length, the stronger the security. However, a greater bit length may decrease performance.

Bit length: 1024

☐ Select cryptographic service provider (CSP) for this certificate


IIS Certificate Wizard

Organization Information

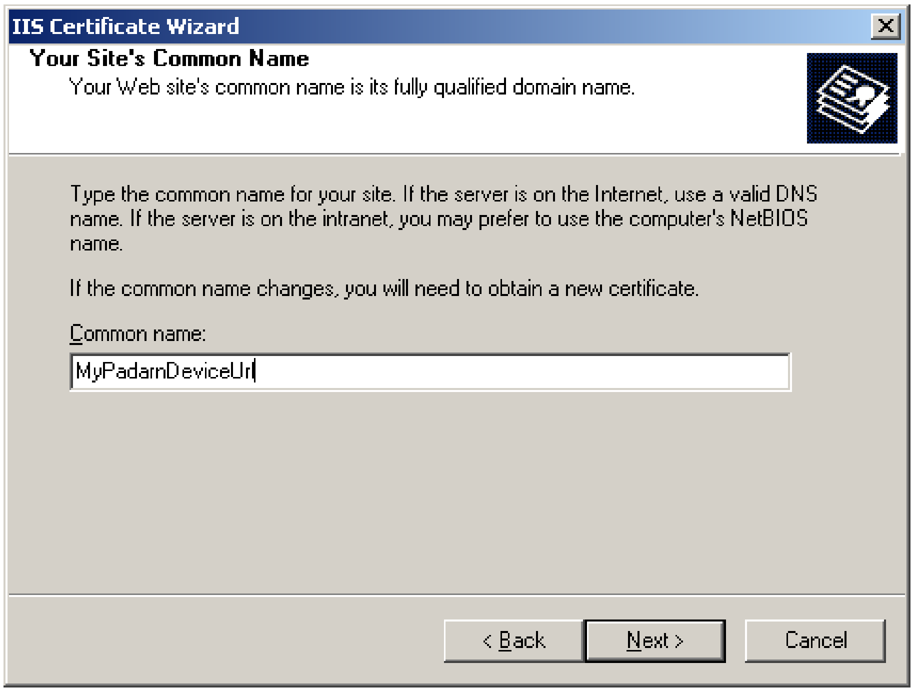
Your certificate must include information about your organization that distinguishes it from other organizations.

Select or type your organization's name and your organizational unit. This is typically the legal name of your organization and the name of your division or department.

For further information, consult certification authority's Web site.

Organization: My Organization

Organizational unit: My Department


IIS Certificate Wizard

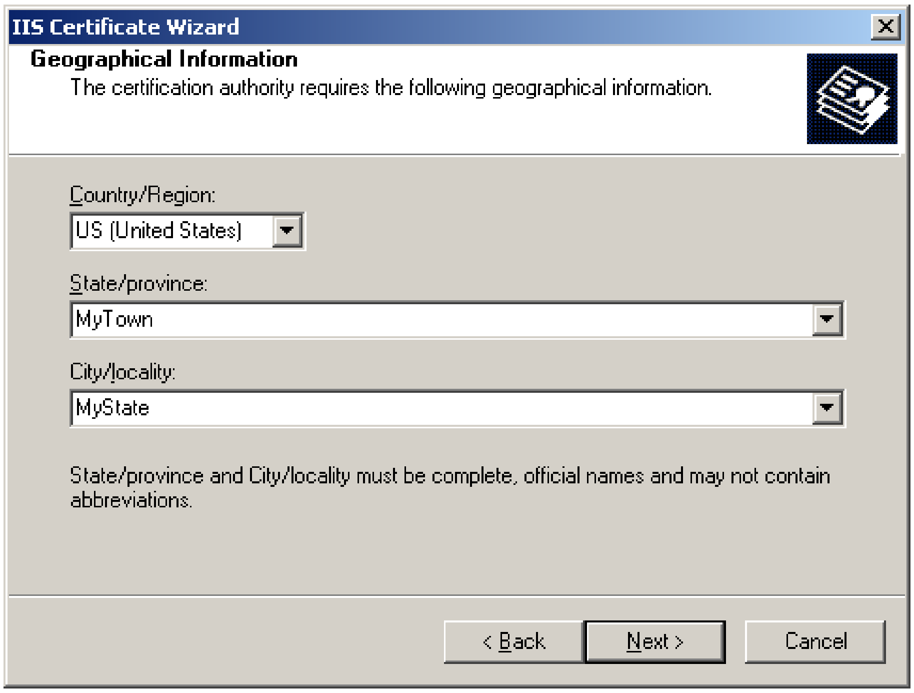
Your Site's Common Name

Your Web site's common name is its fully qualified domain name.

Type the common name for your site. If the server is on the Internet, use a valid DNS name. If the server is on the intranet, you may prefer to use the computer's NetBIOS name.

If the common name changes, you will need to obtain a new certificate.

Common name: MyPadarnDeviceUrl


IIS Certificate Wizard

Geographical Information

The certification authority requires the following geographical information.

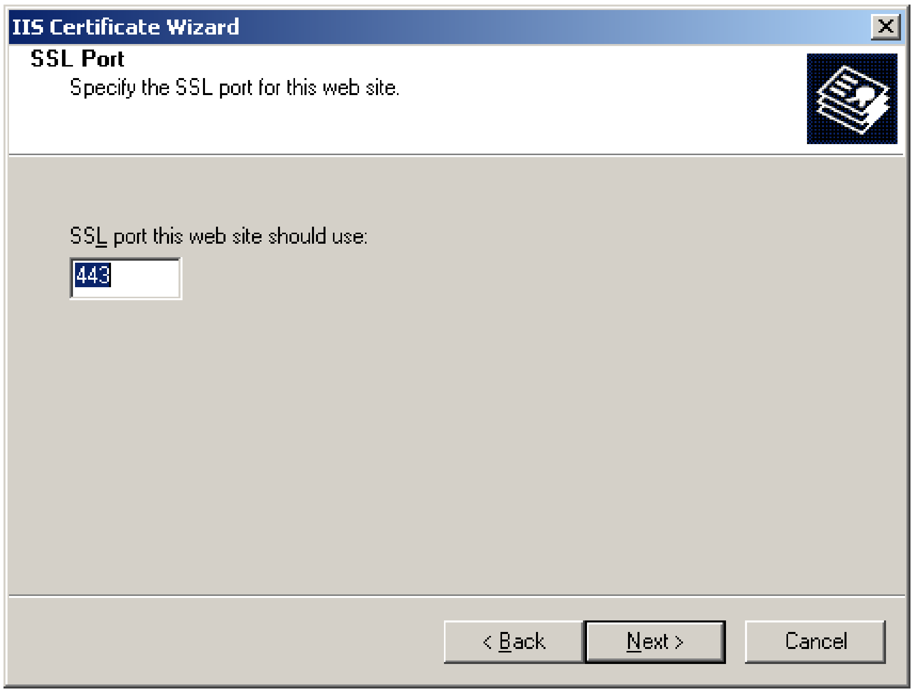
Country/Region: US (United States)

State/province: MyTown

City/locality: MyState

State/province and City/locality must be complete, official names and may not contain abbreviations.

8. In the next four screens, enter in values of your choosing to identify your site.

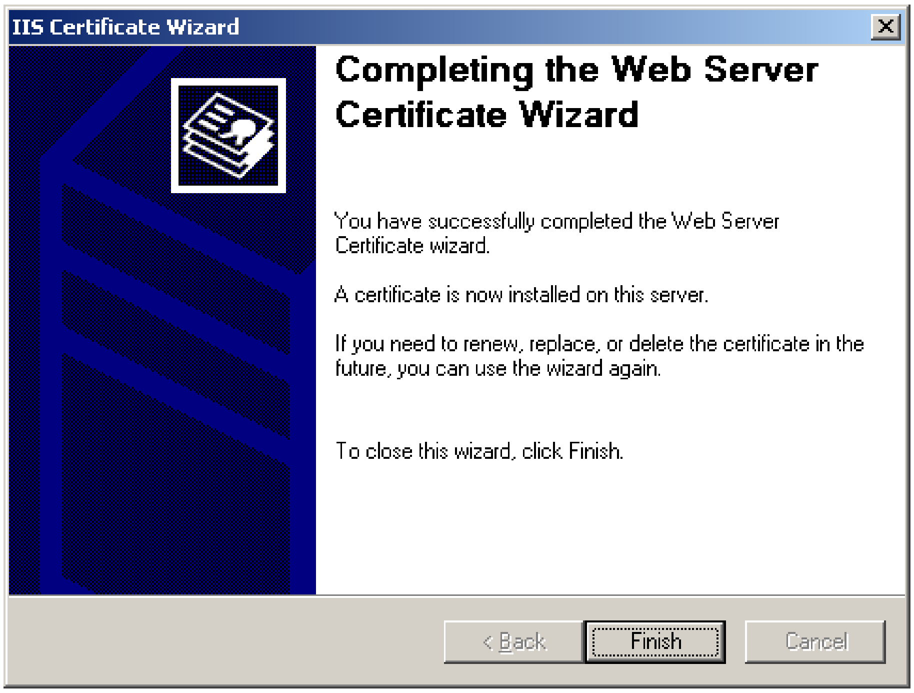

IIS Certificate Wizard

SSL Port

Specify the SSL port for this web site.

SSL port this web site should use: 443

9. Choose 443 as the SSL incoming port.
10. At this point, you can either select your locally-deployed CA server or prepare to send a request to a third party CA.



IIS Certificate Wizard

Completing the Web Server Certificate Wizard

You have successfully completed the Web Server Certificate wizard.

A certificate is now installed on this server.

If you need to renew, replace, or delete the certificate in the future, you can use the wizard again.

To close this wizard, click Finish.

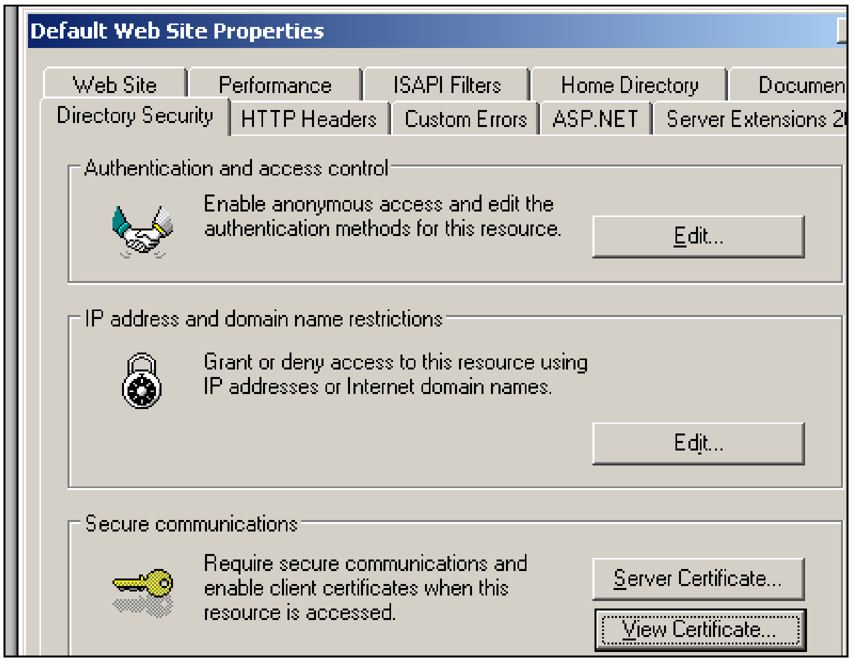
11. The certificate wizard now completes

Exercise 3: Exporting the Personal Information Exchange (PFX) File

A PFX file can be thought of as a container for a piece of sensitive information. In this case, we are going to export the private key of the certificate we just created. Since a private key can be used by a third party to decrypt data coming from our Padarn server, we need to protect this file by applying a strong password. Even though for this lab we are only working with a test certificate, in a production environment, this password would need to be closely guarded (never transmitted in e-mail or written down, for example, beyond the original author).

To create a PFX representation (export) of the certificate we just created:

1. Enter IIS Manager again -> Default Web Site Properties -> Directory Security tab.

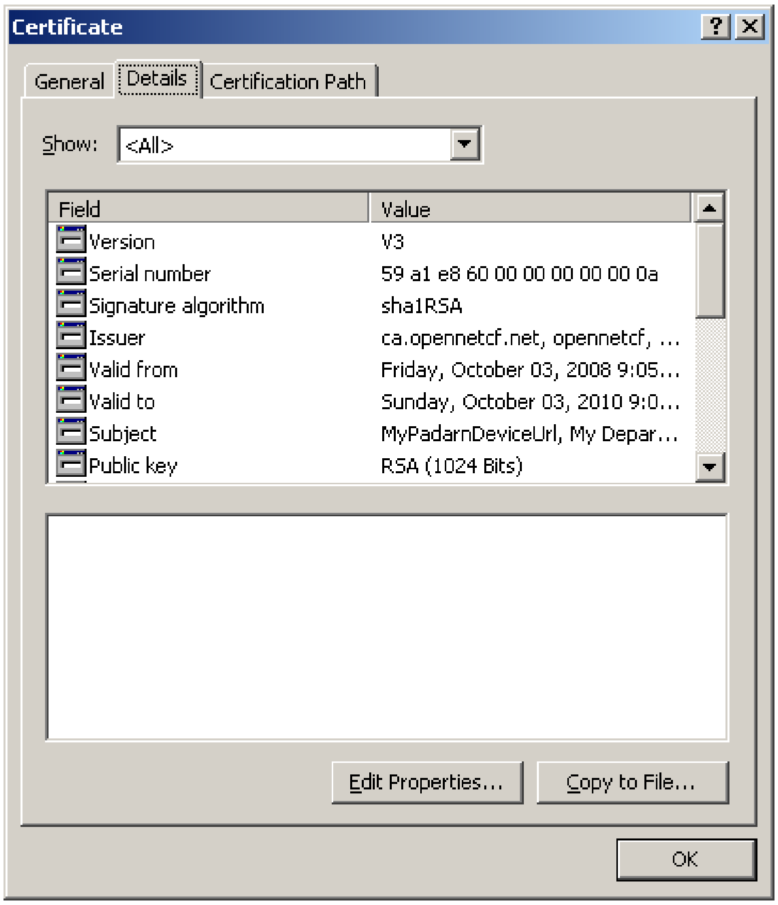


Enable anonymous access and edit the authentication methods for this resource.

Grant or deny access to this resource using IP addresses or Internet domain names.

Require secure communications and enable client certificates when this resource is accessed.

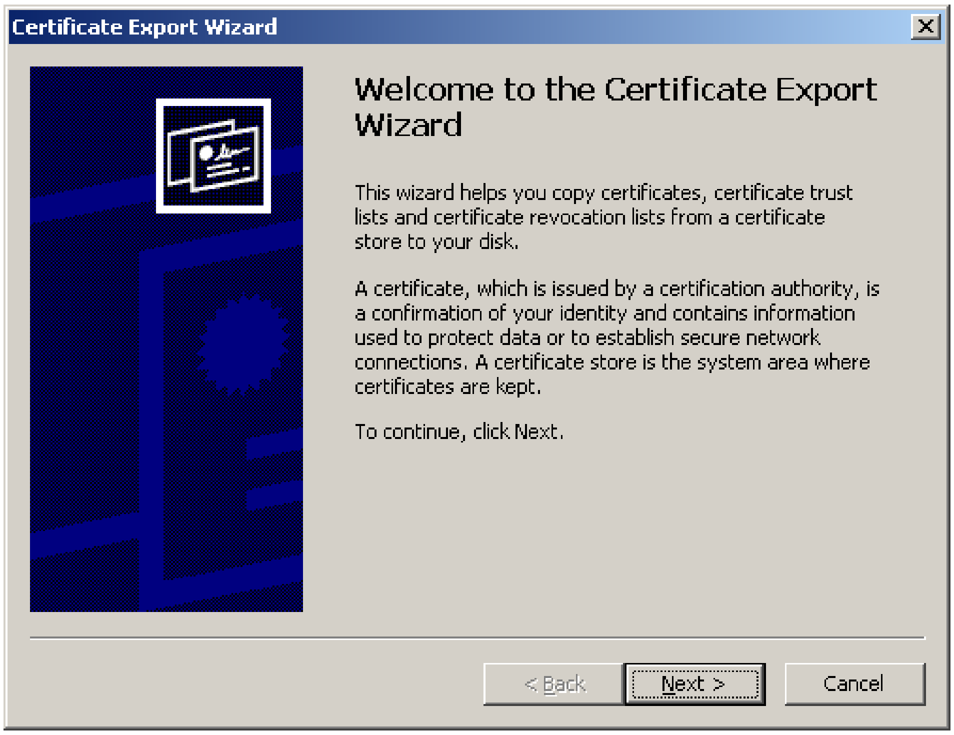
2. Notice the **View Certificate** button is now enabled. Press this button.



Show: <All>

Field	Value
Version	V3
Serial number	59 a1 e8 60 00 00 00 00 00 0a
Signature algorithm	sha1RSA
Issuer	ca.opennetcf.net, opennetcf, ...
Valid from	Friday, October 03, 2008 9:05...
Valid to	Sunday, October 03, 2010 9:0...
Subject	MyPadarnDeviceUrl, My Depar...
Public key	RSA (1024 Bits)

3. Jump into the Details tab.
4. Press **Copy to File**



Certificate Export Wizard

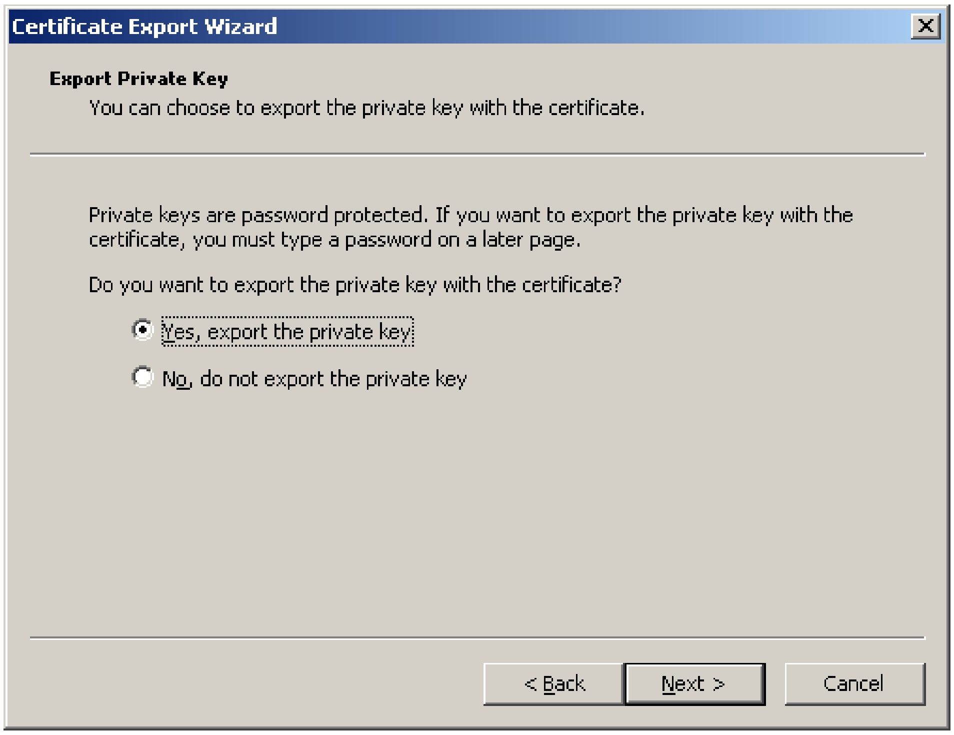
Welcome to the Certificate Export Wizard

This wizard helps you copy certificates, certificate trust lists and certificate revocation lists from a certificate store to your disk.

A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.

To continue, click Next.

5. In the Certificate Export Wizard, press **Next**



Certificate Export Wizard

Export Private Key
You can choose to export the private key with the certificate.

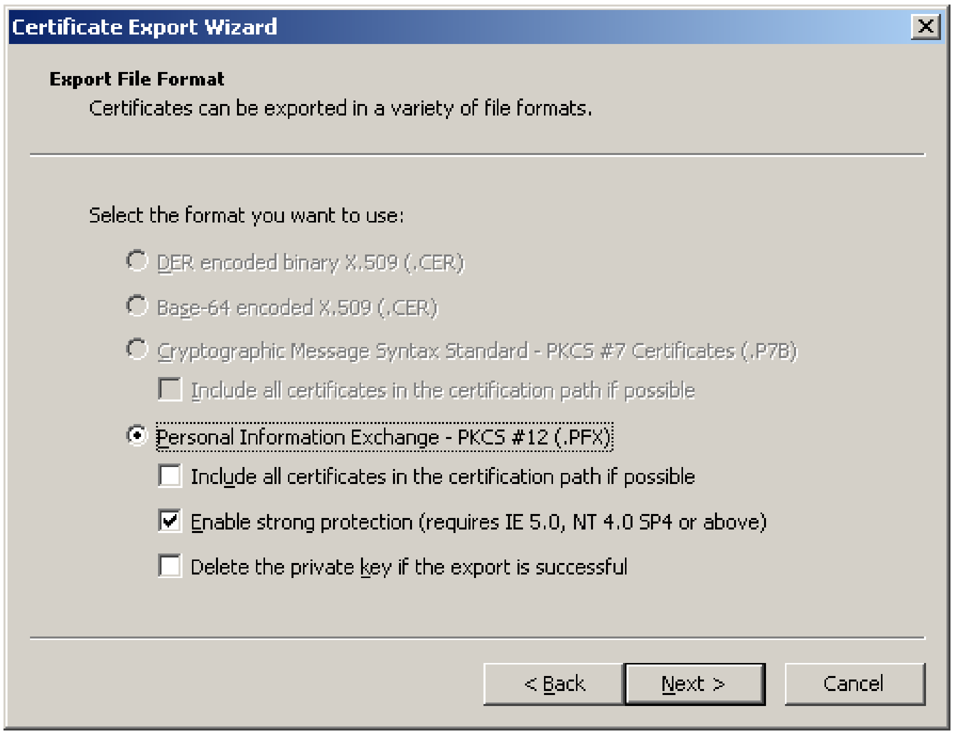
Private keys are password protected. If you want to export the private key with the certificate, you must type a password on a later page.

Do you want to export the private key with the certificate?

[✓ Yes, export the private key]

No, do not export the private key

6. Opt to export the private key

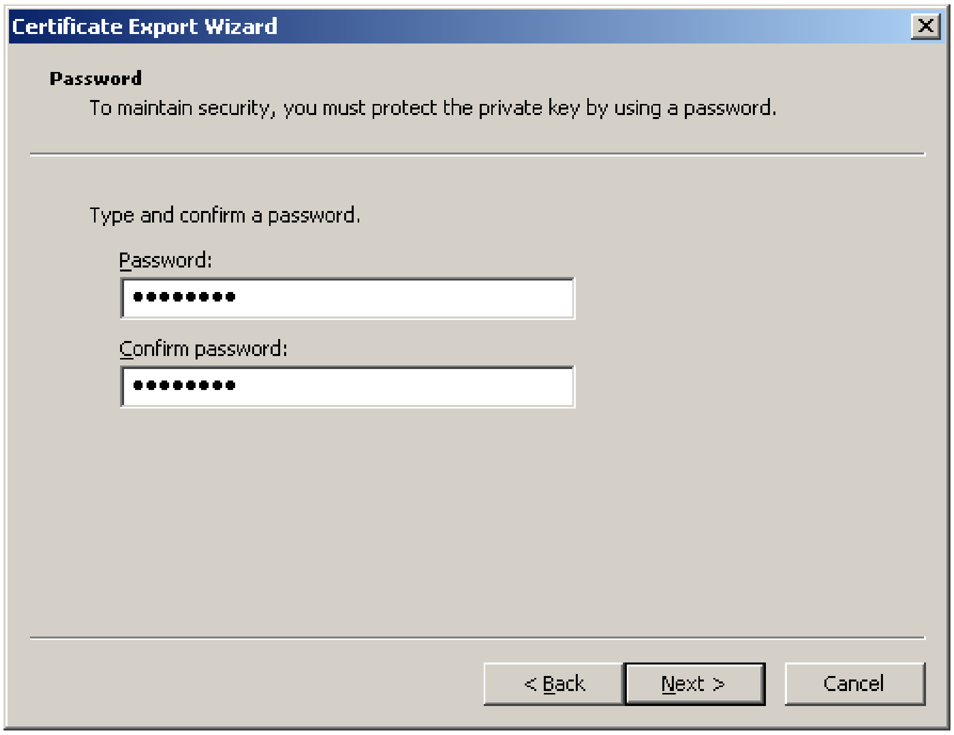

Certificate Export Wizard

Export File Format
Certificates can be exported in a variety of file formats.

Select the format you want to use:

- ( ) DER encoded binary X.509 (.CER)
- ( ) Base-64 encoded X.509 (.CER)
- ( ) Cryptographic Message Syntax Standard - PKCS #7 Certificates (.P7B)
  - [ ] Include all certificates in the certification path if possible
- (●) Personal Information Exchange - PKCS #12 (.PFX)
  - [ ] Include all certificates in the certification path if possible
  - [✓] Enable strong protection (requires IE 5.0, NT 4.0 SP4 or above)
  - [ ] Delete the private key if the export is successful

7. We will use the PKCS #12 format (longer name for PFK). We wish to enable strong protection so the private key doesn't get exposed to unauthorized parties.


Certificate Export Wizard

Password
To maintain security, you must protect the private key by using a password.

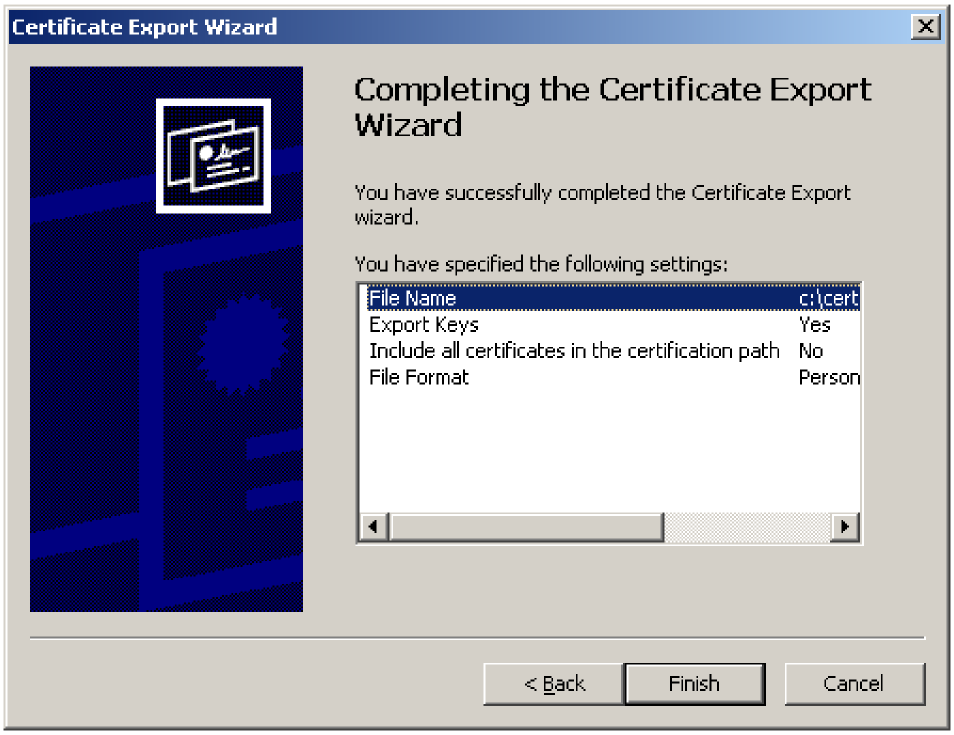
Type and confirm a password.

Password: ●●●●●●●●●

Confirm password: ●●●●●●●●●

8. Be sure to enter a strong password ("4JY!jr"). Always used numbers, letters, and symbols.

9. Name the certificate "server.pfx"



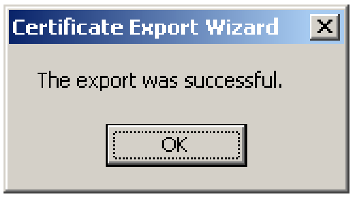
Completing the Certificate Export Wizard

You have successfully completed the Certificate Export wizard.

You have specified the following settings:

File Name	c:\cert
Export Keys	Yes
Include all certificates in the certification path	No
File Format	Person

10. Exit the certificate export wizard by pressing **Finish**



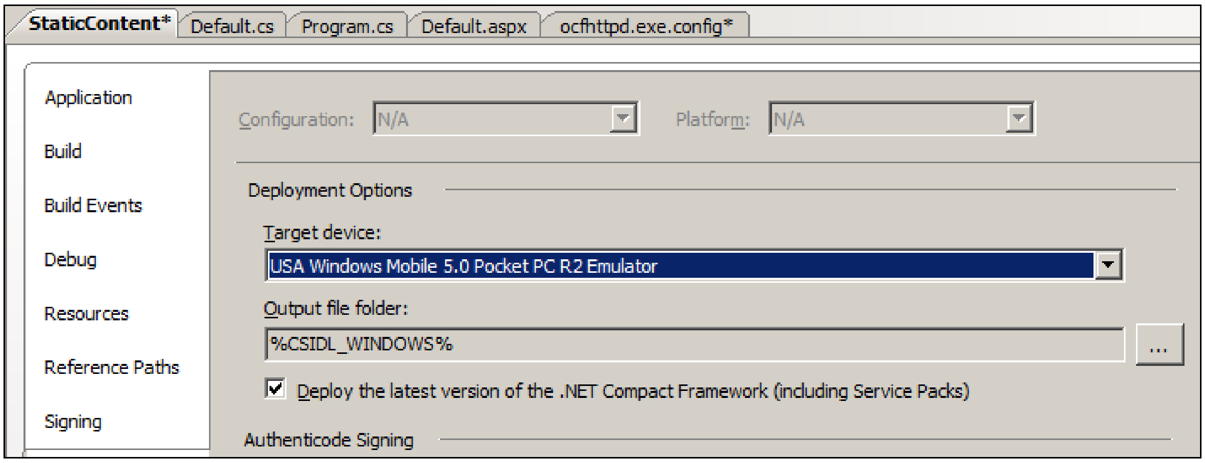
The export was successful.

11. This confirms the exported PFX is valid and ready to be used.

Exercise 4: Add Certificate to Solution for Auto-Deployment

For sake of completeness, we should add the newly-created certificate to our web server solution. To do so:

12. Open the Visual Studio 2008 solution
13. Right click on the StaticContent project and select **Properties -> Devices** and verify that the deployment path is still %CSIDL_WINDOWS%. This verification is important to ensure that any sub-folder items added to this project will be deployed to the appropriately-named sub-folder in the \Windows folder of the target device.



Configuration: N/A
Platform: N/A

Deployment Options

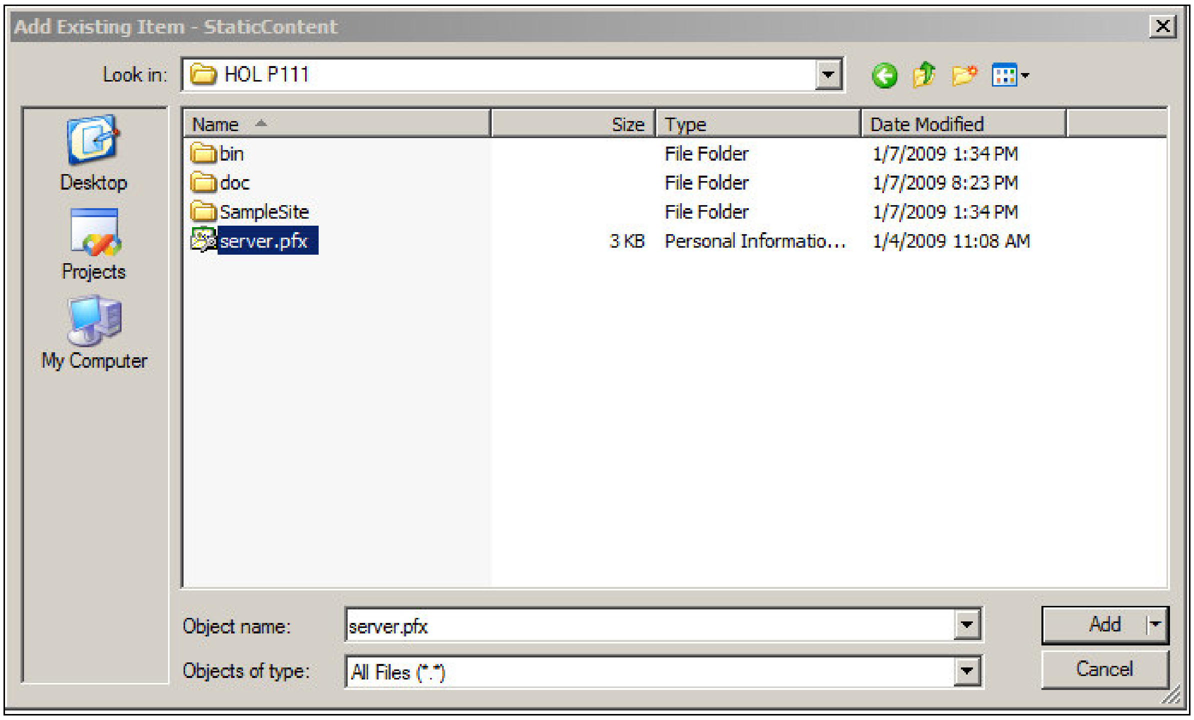
Target device: USA Windows Mobile 5.0 Pocket PC R2 Emulator

Output file folder: %CSIDL_WINDOWS%

☑ Deploy the latest version of the .NET Compact Framework (including Service Packs)

Authenticode Signing

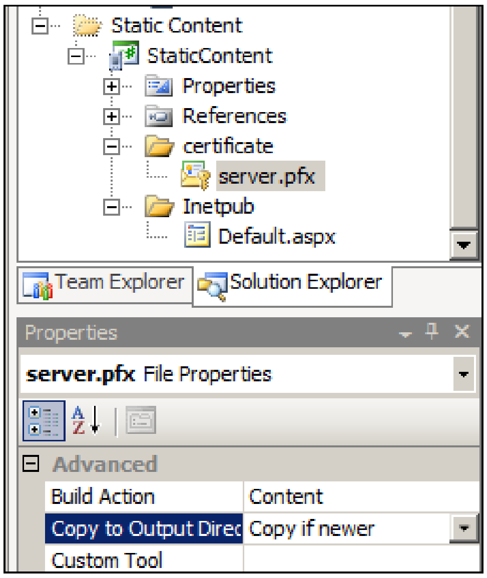
14. Right click on the StaticContent project and choose **Add -> New Folder**
15. Call this folder “certificate”.
16. Add the certificate file generated in exercise 2 to this folder by right clicking on the StaticContent project and choosing **Add Existing Item**.



Name	Size	Type	Date Modified
bin		File Folder	1/7/2009 1:34 PM
doc		File Folder	1/7/2009 8:23 PM
SampleSite		File Folder	1/7/2009 1:34 PM
server.pfx	3 KB	Personal Informatio...	1/4/2009 11:08 AM

Object name: server.pfx
Objects of type: All Files (*.*)

17. Set the build action to “Content” and Copy to Output Directory option to “Copy if Newer”



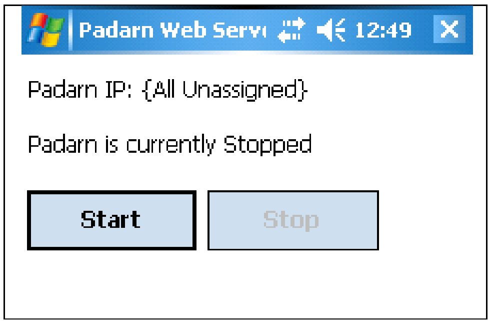
Build Action: Content
Copy to Output Direc: Copy if newer
Custom Tool

That’s it. Now, Visual Studio will always ensure the certificate file exists in \Windows\certificate\server.pfx.

Exercise 5: Verify Digest Authentication

In order to validate that clients will now be exchanging data with Padarn over HTTPS using SSL, , we need to ensure a valid instance of Padarn is up and running; furthermore, you should validate that Default.aspx has been deployed as static content to that Padarn's instance inetpub folder. Instructions for how to accomplish this are in HOL 100.

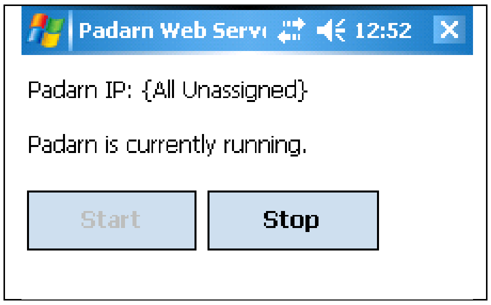
1. Initiate Padarn through the MyPadarnSolution Visual Studio 2008 solution. Right-click on the ocfhttpd project and select **Debug -> Start Without Debugging** (or press Ctrl + F5). For this demonstration, we will be using the Windows Mobile 5.0 Pocket PC R2 Emulator.


Padarn Web Servi 12:49

Padarn IP: {All Unassigned}

Padarn is currently Stopped

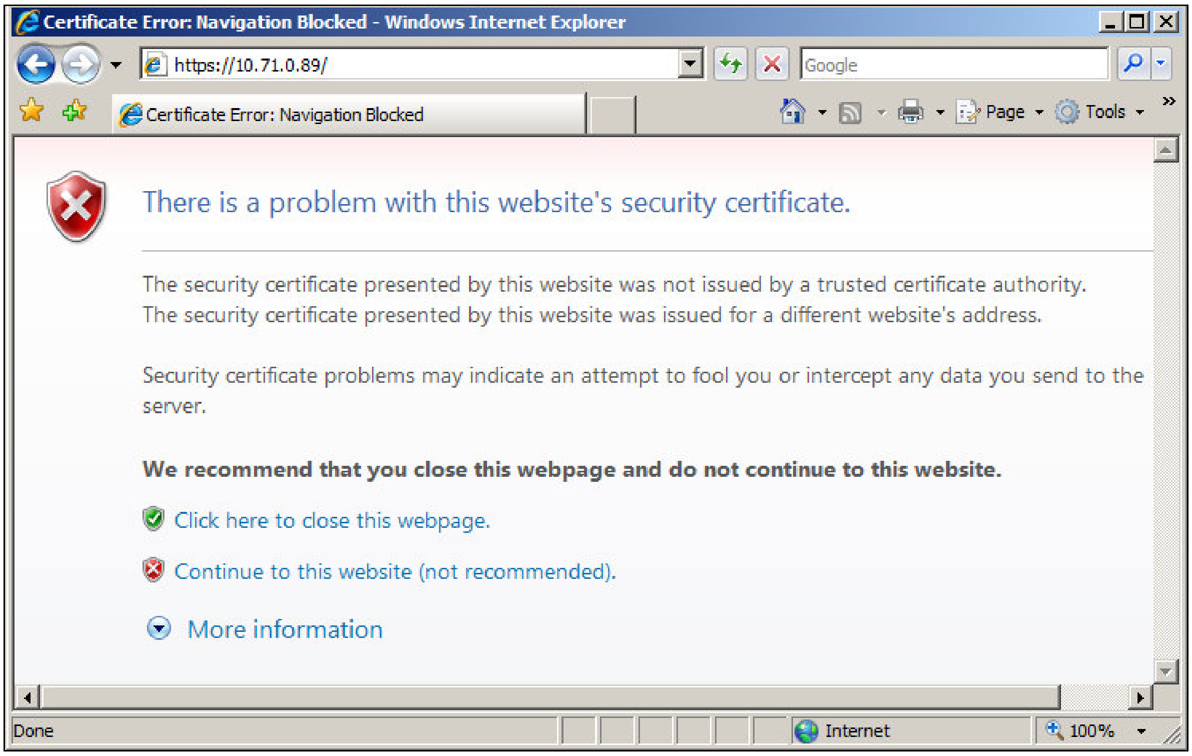
2. Press the **Start** button to initiate the Padarn instance.


Padarn Web Servi 12:52

Padarn IP: {All Unassigned}

Padarn is currently running.

3. From a web browser that can communicate with the Padarn instance (for example, a Windows XP laptop that is host to the Windows Mobile 5.0 emulator). The IP address of the Padarn instance is key for bring up any Padarn-served content. Various utilities exist for Windows CE that provide this information. For example, you could use the OpenNETCF.Net.NetworkInformation namespace.
4. Because we have disabled Basic and Digest Access Authentication for this lab, we will not see a dialog box prompting us for a user name and password. Instead, because we are connecting to a Padarn server using a certificate not published by a reputable CA (unless we decide to use a CA running on the same network as our Padarn server), we will receive a threatening message in our web browser:


Certificate Error: Navigation Blocked - Windows Internet Explorer

https://10.71.0.89/

There is a problem with this website's security certificate.

The security certificate presented by this website was not issued by a trusted certificate authority.
The security certificate presented by this website was issued for a different website's address.

Security certificate problems may indicate an attempt to fool you or intercept any data you send to the server.

We recommend that you close this webpage and do not continue to this website.

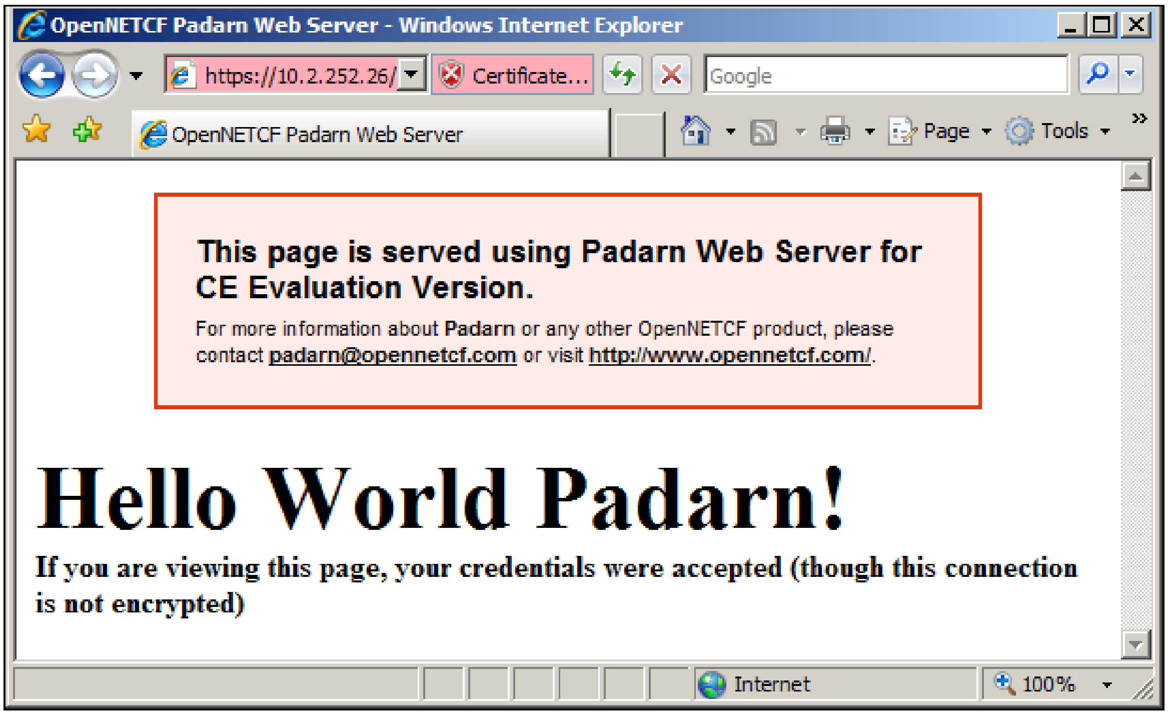
Click here to close this webpage.

Continue to this website (not recommended).

More information

Done

5. Click "Continue to this website (not recommended)" since we know we're visiting our Padarn server


OpenNETCF Padarn Web Server - Windows Internet Explorer

https://10.2.252.26/ Certificate...

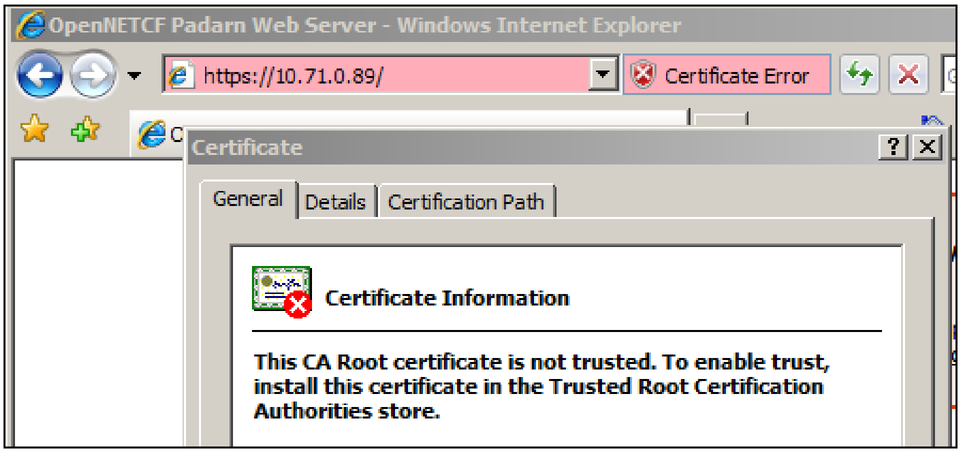
This page is served using Padarn Web Server for CE Evaluation Version.

For more information about **Padarn** or any other OpenNETCF product, please contact padarn@opennetcf.com or visit http://www.opennetcf.com/.

Hello World Padarn!

If you are viewing this page, your credentials were accepted (though this connection is not encrypted)

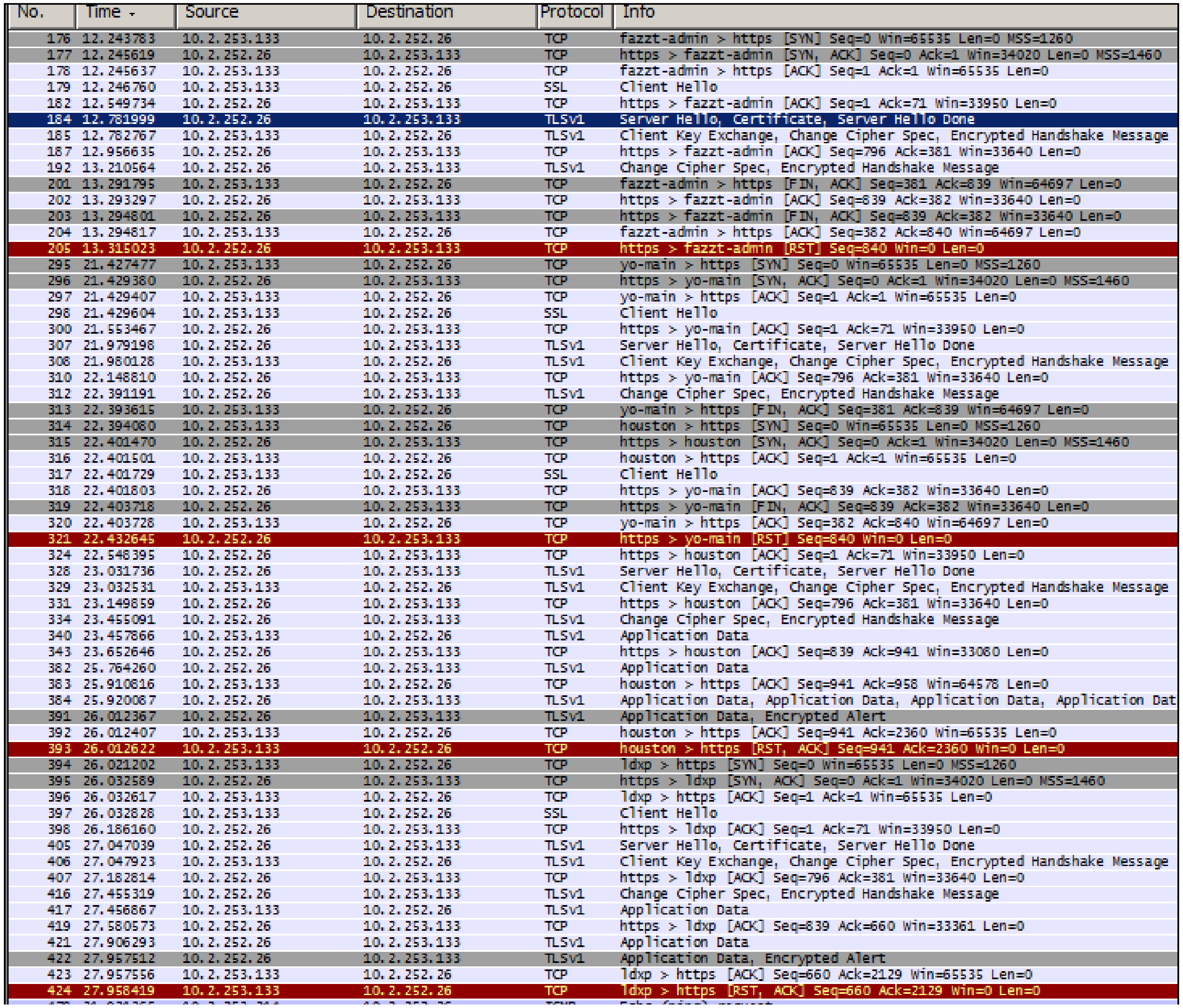
6. Confirm the "Hello World Padarn" website appears, though with certificate errors:



This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.

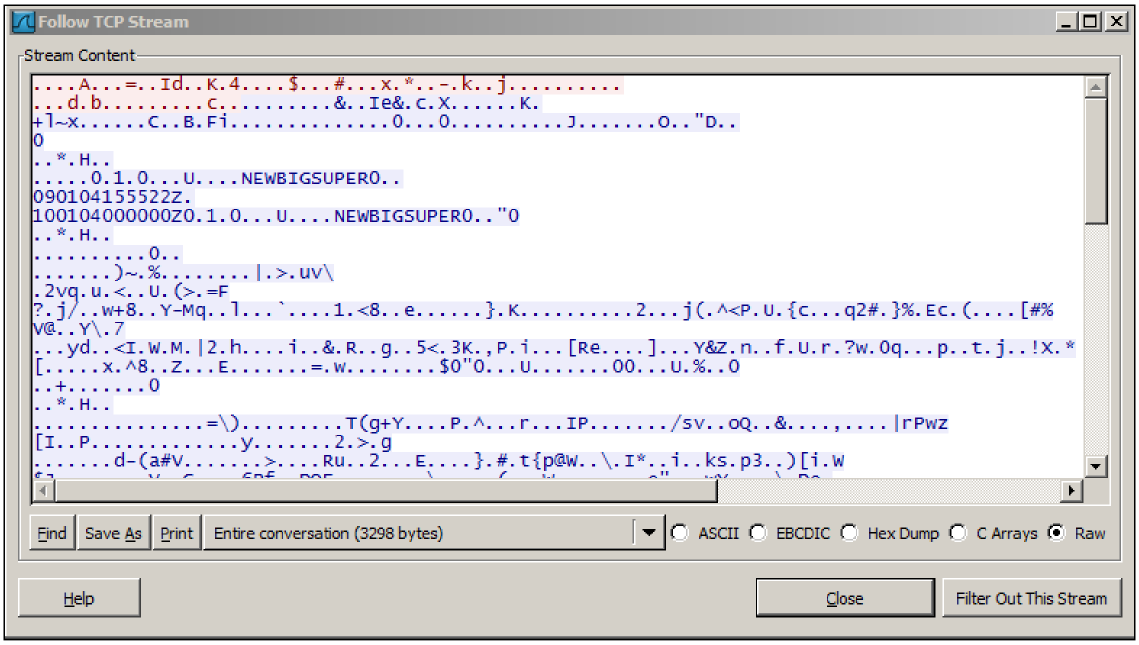
We can use a network sniffer tool such as Wireshark ([information here]) to determine all headers exchanged between Padarn and the desktop instance. Here is a brief snapshot of what we would expect to see on the serv

OpenNETCF Consulting



No.	Time	Source	Destination	Protocol	Info
176	12.243783	10.2.253.133	10.2.252.26	TCP	fazzt-admin > https [SYN] Seq=0 Win=65535 Len=0 MSS=1260
177	12.245619	10.2.252.26	10.2.253.133	TCP	https > fazzt-admin [SYN, ACK] Seq=0 Ack=1 Win=34020 Len=0 MSS=1460
178	12.245637	10.2.253.133	10.2.252.26	TCP	fazzt-admin > https [ACK] Seq=1 Ack=1 Win=65535 Len=0
179	12.246760	10.2.253.133	10.2.252.26	SSL	Client Hello
182	12.549734	10.2.252.26	10.2.253.133	TCP	https > fazzt-admin [ACK] Seq=1 Ack=71 Win=33950 Len=0
184	12.781999	10.2.252.26	10.2.253.133	TLSv1	Server Hello, Certificate, Server Hello Done
185	12.782767	10.2.253.133	10.2.252.26	TLSv1	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
187	12.956635	10.2.252.26	10.2.253.133	TCP	https > fazzt-admin [ACK] Seq=796 Ack=381 Win=33640 Len=0
192	13.210564	10.2.252.26	10.2.253.133	TLSv1	Change Cipher Spec, Encrypted Handshake Message
201	13.291795	10.2.253.133	10.2.252.26	TCP	fazzt-admin > https [FIN, ACK] Seq=381 Ack=839 Win=64697 Len=0
202	13.293297	10.2.252.26	10.2.253.133	TCP	https > fazzt-admin [ACK] Seq=839 Ack=382 Win=33640 Len=0
203	13.294801	10.2.252.26	10.2.253.133	TCP	https > fazzt-admin [FIN, ACK] Seq=839 Ack=382 Win=33640 Len=0
204	13.294817	10.2.253.133	10.2.252.26	TCP	fazzt-admin > https [ACK] Seq=382 Ack=840 Win=64697 Len=0
205	13.315023	10.2.252.26	10.2.253.133	TCP	https > fazzt-admin [RST] Seq=840 Win=0 Len=0
295	21.427477	10.2.253.133	10.2.252.26	TCP	yo-main > https [SYN] Seq=0 Win=65535 Len=0 MSS=1260
296	21.429380	10.2.252.26	10.2.253.133	TCP	https > yo-main [SYN, ACK] Seq=0 Ack=1 Win=34020 Len=0 MSS=1460
297	21.429407	10.2.253.133	10.2.252.26	TCP	yo-main > https [ACK] Seq=1 Ack=1 Win=65535 Len=0
298	21.429604	10.2.253.133	10.2.252.26	SSL	Client Hello
300	21.553467	10.2.252.26	10.2.253.133	TCP	https > yo-main [ACK] Seq=1 Ack=71 Win=33950 Len=0
307	21.979198	10.2.252.26	10.2.253.133	TLSv1	Server Hello, Certificate, Server Hello Done
308	21.980128	10.2.253.133	10.2.252.26	TLSv1	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
310	22.148810	10.2.252.26	10.2.253.133	TCP	https > yo-main [ACK] Seq=796 Ack=381 Win=33640 Len=0
312	22.391191	10.2.252.26	10.2.253.133	TLSv1	Change Cipher Spec, Encrypted Handshake Message
313	22.393615	10.2.253.133	10.2.252.26	TCP	yo-main > https [FIN, ACK] Seq=381 Ack=839 Win=64697 Len=0
314	22.394080	10.2.253.133	10.2.252.26	TCP	houston > https [SYN] Seq=0 Win=65535 Len=0 MSS=1260
315	22.401470	10.2.252.26	10.2.253.133	TCP	https > houston [SYN, ACK] Seq=0 Ack=1 Win=34020 Len=0 MSS=1460
316	22.401501	10.2.253.133	10.2.252.26	TCP	houston > https [ACK] Seq=1 Ack=1 Win=65535 Len=0
317	22.401729	10.2.253.133	10.2.252.26	SSL	Client Hello
318	22.401803	10.2.252.26	10.2.253.133	TCP	https > yo-main [ACK] Seq=839 Ack=382 Win=33640 Len=0
319	22.403718	10.2.252.26	10.2.253.133	TCP	https > yo-main [FIN, ACK] Seq=839 Ack=382 Win=33640 Len=0
320	22.403728	10.2.253.133	10.2.252.26	TCP	yo-main > https [ACK] Seq=382 Ack=840 Win=64697 Len=0
321	22.432645	10.2.252.26	10.2.253.133	TCP	https > yo-main [RST] Seq=840 Win=0 Len=0
324	22.548395	10.2.252.26	10.2.253.133	TCP	https > houston [ACK] Seq=1 Ack=71 Win=33950 Len=0
328	23.031736	10.2.252.26	10.2.253.133	TLSv1	Server Hello, Certificate, Server Hello Done
329	23.032531	10.2.253.133	10.2.252.26	TLSv1	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
331	23.149859	10.2.252.26	10.2.253.133	TCP	https > houston [ACK] Seq=796 Ack=381 Win=33640 Len=0
334	23.455091	10.2.252.26	10.2.253.133	TLSv1	Change Cipher Spec, Encrypted Handshake Message
340	23.457866	10.2.253.133	10.2.252.26	TLSv1	Application Data
343	23.652646	10.2.252.26	10.2.253.133	TCP	https > houston [ACK] Seq=839 Ack=941 Win=33080 Len=0
382	25.764260	10.2.252.26	10.2.253.133	TLSv1	Application Data
383	25.910816	10.2.253.133	10.2.252.26	TCP	houston > https [ACK] Seq=941 Ack=958 Win=64578 Len=0
384	25.920087	10.2.252.26	10.2.253.133	TLSv1	Application Data, Application Data, Application Data, Application Dat
391	26.012367	10.2.252.26	10.2.253.133	TLSv1	Application Data, Encrypted Alert
392	26.012407	10.2.253.133	10.2.252.26	TCP	houston > https [ACK] Seq=941 Ack=2360 Win=65535 Len=0
393	26.012622	10.2.253.133	10.2.252.26	TCP	houston > https [RST, ACK] Seq=941 Ack=2360 Win=0 Len=0
394	26.021202	10.2.253.133	10.2.252.26	TCP	ldxp > https [SYN] Seq=0 Win=65535 Len=0 MSS=1260
395	26.032589	10.2.252.26	10.2.253.133	TCP	https > ldxp [SYN, ACK] Seq=0 Ack=1 Win=34020 Len=0 MSS=1460
396	26.032617	10.2.253.133	10.2.252.26	TCP	ldxp > https [ACK] Seq=1 Ack=1 Win=65535 Len=0
397	26.032828	10.2.253.133	10.2.252.26	SSL	Client Hello
398	26.186160	10.2.252.26	10.2.253.133	TCP	https > ldxp [ACK] Seq=1 Ack=71 Win=33950 Len=0
405	27.047039	10.2.252.26	10.2.253.133	TLSv1	Server Hello, Certificate, Server Hello Done
406	27.047923	10.2.253.133	10.2.252.26	TLSv1	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
407	27.182814	10.2.252.26	10.2.253.133	TCP	https > ldxp [ACK] Seq=796 Ack=381 Win=33640 Len=0
416	27.455319	10.2.252.26	10.2.253.133	TLSv1	Change Cipher Spec, Encrypted Handshake Message
417	27.456867	10.2.253.133	10.2.252.26	TLSv1	Application Data
419	27.580573	10.2.252.26	10.2.253.133	TCP	https > ldxp [ACK] Seq=839 Ack=660 Win=33361 Len=0
421	27.906293	10.2.252.26	10.2.253.133	TLSv1	Application Data
422	27.957512	10.2.252.26	10.2.253.133	TLSv1	Application Data, Encrypted Alert
423	27.957556	10.2.253.133	10.2.252.26	TCP	ldxp > https [ACK] Seq=660 Ack=2129 Win=65535 Len=0
424	27.958419	10.2.253.133	10.2.252.26	TCP	ldxp > https [RST, ACK] Seq=660 Ack=2129 Win=0 Len=0

Notice in the stream above (this covers *only* the SSL handshake as described in the introduction to this lab; it does not include credential authentication!) the client and server "Hello" messages, followed by the Server certificate details and the Server "Hello Done" message. The handshake is actually performed three full times because the PFK used by Padarn is invalid and not trusted by the client (in this case, Internet Explorer 7).



```
....A...=..Id..K.4....$...#...x.*..-.k..j..........
...d.b.........c..........&..Ie&.c.X......K.
+l~x......C..B.Fi..............0...0..........J.......0.."D..
0
..*.H..
.....0.1.0...U....NEWBIGSUPER0..
090104155522Z.
100104000000Z0.1.0...U....NEWBIGSUPER0.."0
..*.H..
..........0..
.......)~.%........|.>.uv\
.2vq.u.<..U.(>.=F
?.j/..w+8..Y-Mq..l...`....1.<8..e......}.K..........2...j(.^<P.U.{c...q2#.}%.Ec.(....[#%
V@..Y\.7
...yd..<I.W.M.|2.h....i..&.R..g..5<.3K.,P.i...[Re....]...Y&Z.n..f.U.r.?w.0q...p..t.j..!X.*
[.....x.^8..Z...E........=.w.........$0"0...U.......00...U.%..0
..+.......0
..*.H..
...............=\).........T(g+Y....P.^...r...IP......./sv..oQ..&....,....|rPwz
[I..P.............y.......2.>.g
.......d-(a#V.......>....Ru..2...E....}.#.t{p@w..\.I*..i..ks.p3..)[i.w
```

If you try to follow one of the "Application Data" TLSV1 streams, you'll see the above. Just a bunch of encrypted data!

SSL might seem like a boon for web server (and client information) protection, but there are some drawbacks:

1. SSL requires a certificate. Working with a CA is a maintenance task and costs money
2. Every packet transferred between server and client is encrypted. For mobile devices, this has a larger toll:
    a. Size of packets is greater (of concern with devices using pay-per-byte)
    b. Encryption of packets requires additional processing time
    c. Decryption of packets requires additional processing time

Hands-on Lab Summary

SSL over HTTPS provides industry-standard encryption plus server authentication that will allow you to trust Padarn with sensitive data. Once a CA has been contacted and a certificate file created on your behalf, it is quite straightforward to configure Padarn to require all clients to abide by the rules of an SSL exchange. Any non-participating clients will be refused access.

Although SSL is more expensive both in terms of network consumption and processor usage than Digest Authentication, for transactions that involve confidential data, SSL should be your top choice for server <-> client communication protection.

In this lab you:

- ✓ Re-used an existing your Padarn website solution
- ✓ Made a simple change to the Padarn ocfhttpd.exe.config to enable SSL transactions
- ✓ Added a Windows Server-compatible Personal Information Exchange (PFX) certificate export to Padarn
- ✓ Re-used a simple "Hello World" style test landing page for verification of SSL in action
- ✓ Stepped through the SSL "handshake" within a network packet analyzer to verify encryption of all data exchanged

Hopefully, this lab has convinced you that SSL support built into Padarn v1.1 will allow you to handle even sensitive data exchanges with clients.