**HOL P111 – Securing a Padarn Site with SSL**

# Table of Contents

[Table of Contents 2](#_Toc219129016)

[HOL Requirements 3](#_Toc219129017)

[Summary 4](#_Toc219129018)

[Lab Objective 4](#_Toc219129019)

[Background 4](#_Toc219129020)

[Exercise 1: Configuring Padarn to use SSL Authentication 7](#_Toc219129021)

[Exercise 2: Creating a test certificate 9](#_Toc219129022)

[Exercise 3: Exporting the Personal Information Exchange (PFX) File 16](#_Toc219129023)

[Exercise 4: Add Certificate to Solution for Auto-Deployment 21](#_Toc219129024)

[Exercise 5: Verify Digest Authentication 23](#_Toc219129025)

[Hands-on Lab Summary 28](#_Toc219129026)

# 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.

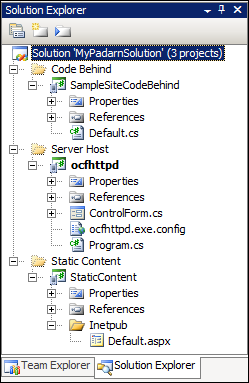


At this point, the handshake process has outputted a key that will be used for all subsequent exchanges during this session. It’s important to realize that the key is not just generic/random salt – it’s based upon agreed upon values such as identity and SSL version. In more detail:

1. 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.
2. 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.
3. The client now tells Padarn (in an encrypted message) that it is ready to proceed using the master secret
4. Padarn now tells the client that it is ready to begin using the master secret
5. 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.

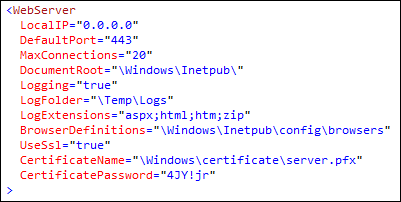
## Exercise 1: Configuring Padarn to use SSL Authentication

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:

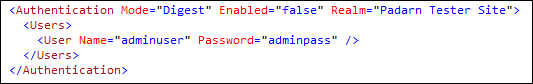


All security settings pertaining to a given Padarn instance exist within ocfhttpd.exe.config. The file exists within the ocfhttpd project already. There are two sections where we will focus our attention.

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



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.



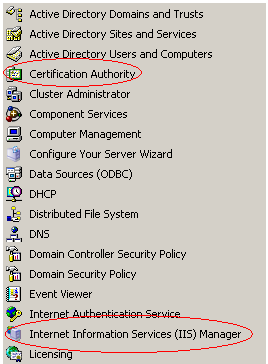
For simplicity sake, we will not be using an additional form of authentication; however, it is clearly a best practice to secure any website deemed worthy of encryption with an entry user name and password! Padarn supports both Basic and Digest Access Authentication over SSL.

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**.

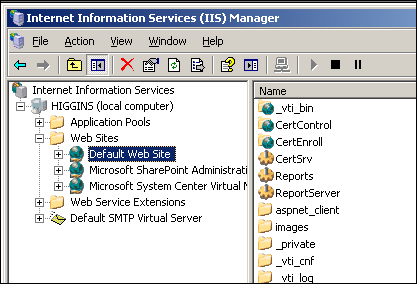
## Exercise 2: Creating a test certificate

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:

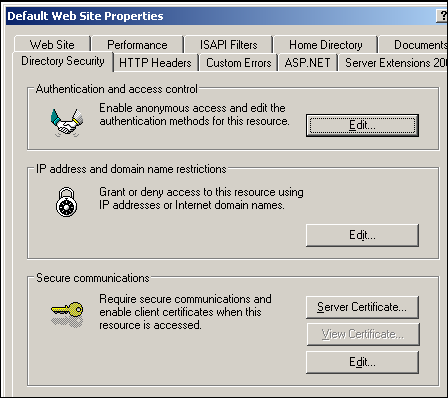
1. Open your Administrative Tools menu on your server and verify the Certification Authority and Internet Information Services (IIS Manager) are installed.



1. Open IIS Manager



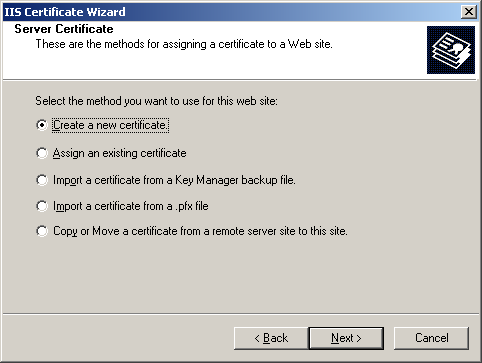
1. Right click Default Web Site (the website configured by IIS installation) with default content available



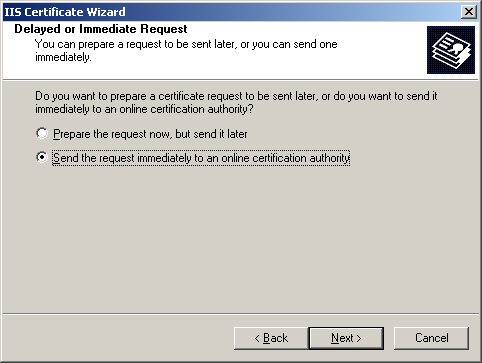
1. Press **Server Certificate**



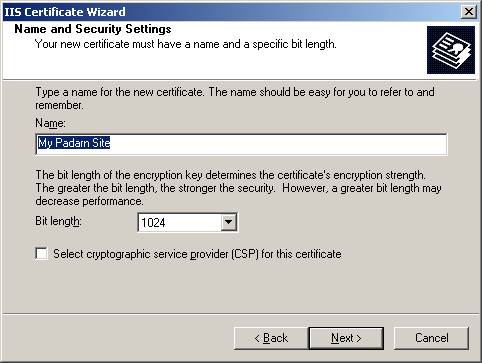
1. Press Next

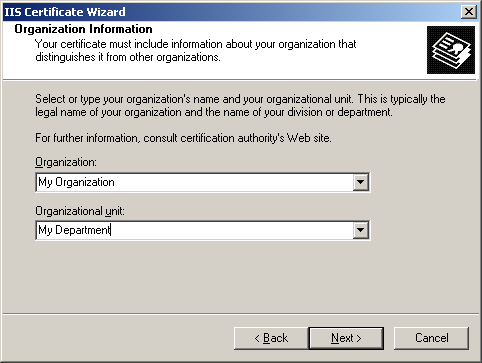


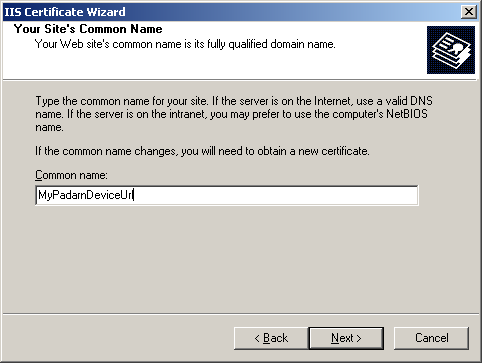
1. Select “Create a new certificate” and select **Next**

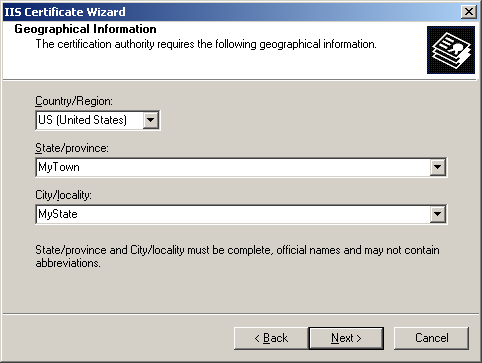


1. On the Delayed/Immediate Request dialog, select either choice. Sending a request to a real third party CA is beyond the scope of this lab.

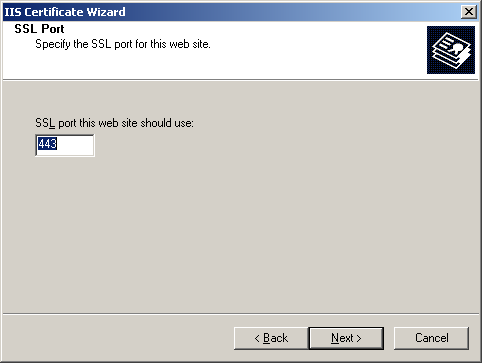




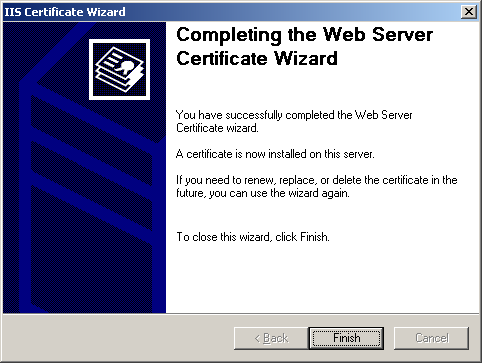




1. In the next four screens, enter in values of your choosing to identify your site.



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



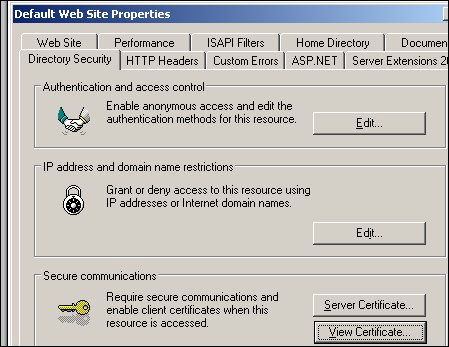
1. The certificate wizard now completes

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

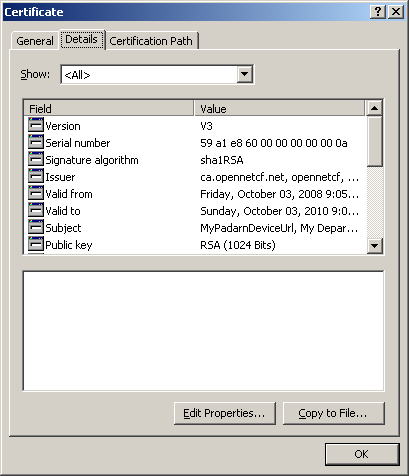
A PFX file can be thought of as a container for a piece of sensitive information. In this case, we are going to export the private key of the certificate we just created. Since a private key can be used by a third party to decrypt data coming from our Padarn server, we need to protect this file by applying 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.



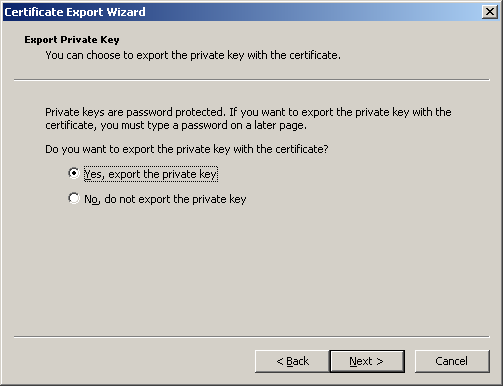
1. Notice the **View Certificate** button is now enabled. Press this button.



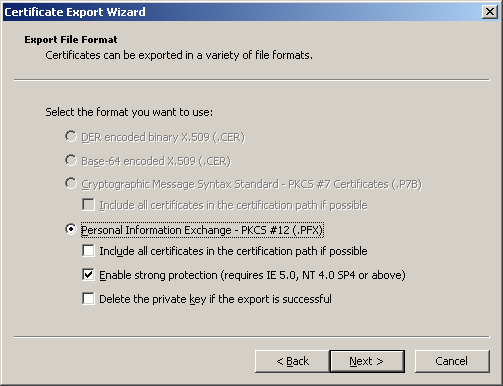
1. Jump into the Details tab.
2. Press **Copy to File**



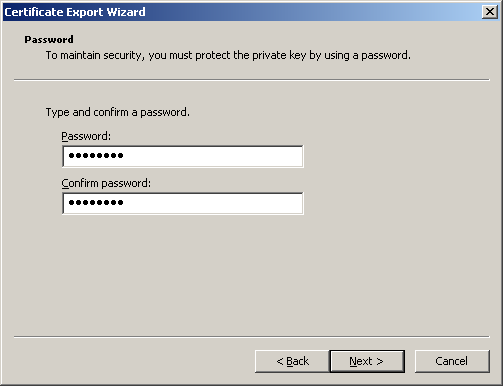
1. In the Certificate Export Wizard, press **Next**



1. Opt to export the private key



1. 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.



1. Be sure to enter a strong password (“4JY!jr”). Always used numbers, letters, and symbols.
2. Name the certificate “server.pfx”



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

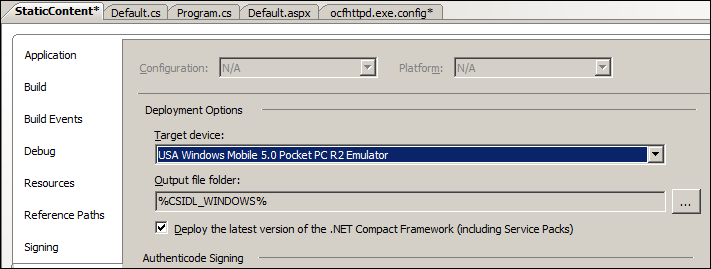


1. This confirms the exported PFX is valid and ready to be used.

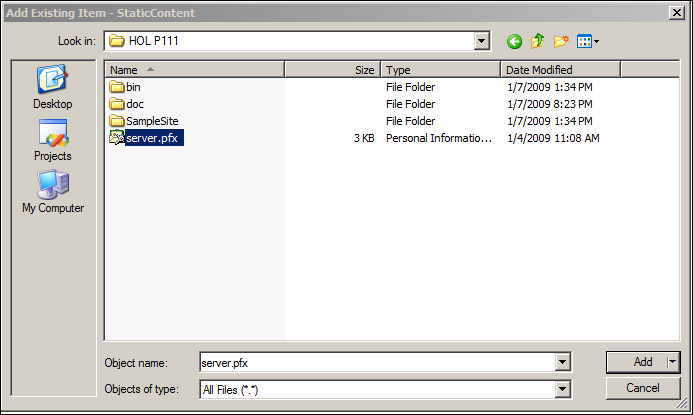
## Exercise 4: Add Certificate to Solution for Auto-Deployment

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

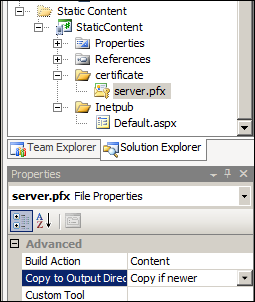
1. Open the Visual Studio 2008 solution
2. Right click on the StaticContent project and select **Properties -> Devices** and verify that the deployment path is still %CSIDL\_WINDOWS%. This verification is important to ensure that any sub-folder items added to this project will be deployed to the appropriately-named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



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



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



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.

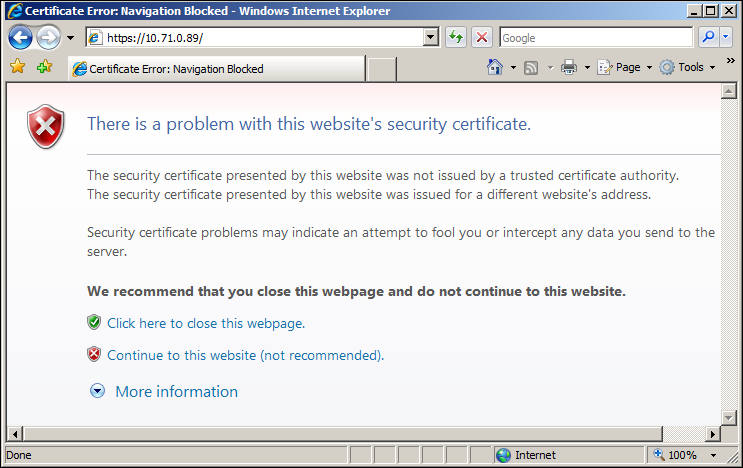
1. Initiate Padarn through the MyPadarnSolution Visual Studio 2008 solution. Right-click on the ocfhttpd project and select **Debug -> Start Without Debugging** (or press Ctrl + F5). For this demonstration, we will be using the Windows Mobile 5.0 Pocket PC R2 Emulator.

|  |
| --- |
|  |

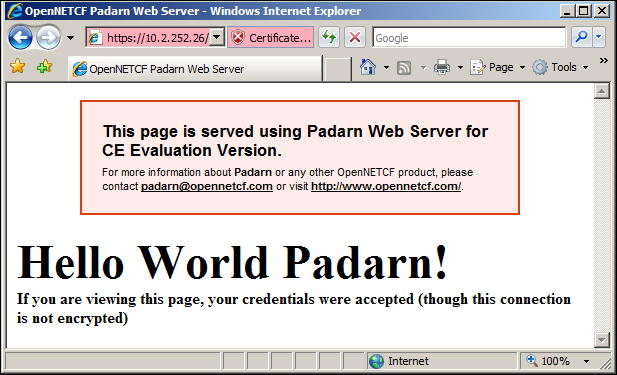
1. Press the **Start** button to initiate the Padarn instance.

|  |
| --- |
|  |

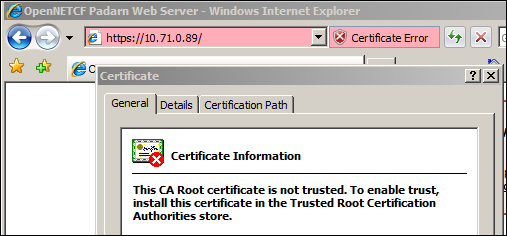
1. From a web browser that can communicate with the Padarn instance (for example, a Windows XP laptop that is host to the Windows Mobile 5.0 emulator). The IP address of the Padarn instance is key for bring up any Padarn-served content. Various utilities exist for Windows CE that provide this information. For example, you could use the [OpenNETCF.Net.NetworkInformation](http://www.opennetcf.com/library/sdf/html/d79a4a4f-93be-3c85-477b-5298d1141978.htm) namespace.
2. 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:



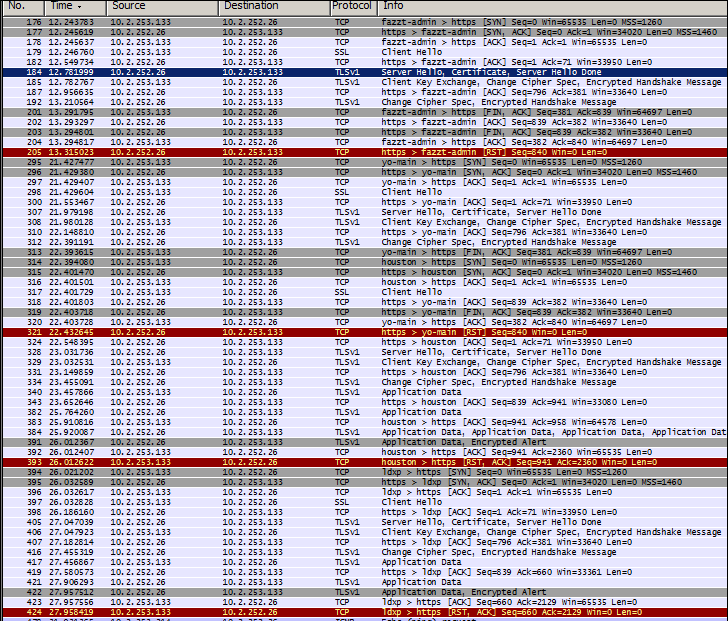
1. Click “Continue to this website (not recommended)” since we know we’re visiting our Padarn server



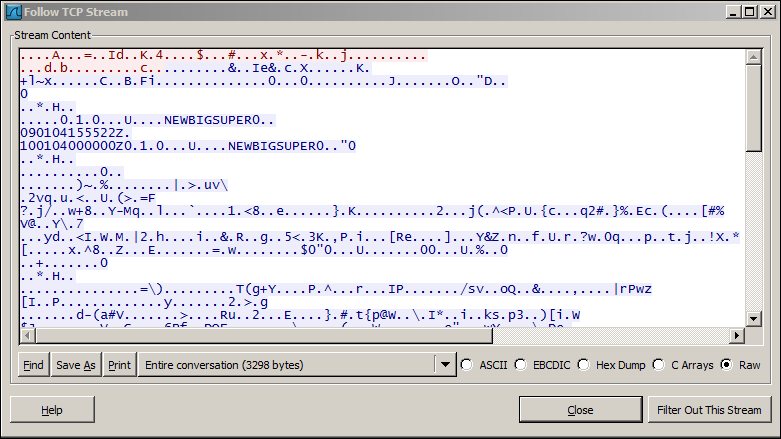
1. Confirm the “Hello World Padarn” website appears, though with certificate errors:



We can use a network sniffer tool such as Wireshark ([information here](http://www.wireshark.org/)) 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



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).



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:
   1. Size of packets is greater (of concern with devices using pay-per-byte)
   2. Encryption of packets requires additional processing time
   3. 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.