

HOL P110 – Securing a Padarn Site with Digest Authentication

Table of Contents

Table of Contents	2
HOL Requirements	3
Summary.....	4
Lab Objective	4
Background.....	4
Exercise 1: Configuring a basic Padarn web resource	7
Exercise 2: Configuring Padarn to use Digest Access Authentication	12
Exercise 3: Verify Digest Authentication	13
Hands-on Lab Summary	17

HOL Requirements

The following items are required to run this HOL:

- A Desktop PC running Microsoft Windows® XP or Windows Vista
- 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 Digest Access Authentication including why it is an improvement over Basic Authentication, details as to how Padarn implements Digest Access Authentication, and finally how to configure a Padarn web server instance to use Digest Access Authentication.

Lab Objective

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

In this HOL, you will perform the following exercises:

- ✓ Learn how to modify the `ocfhttpd.exe.config` file to enable Digest Access Authentication
- ✓ Test to ensure Digest Access Authentication is being used through a live request

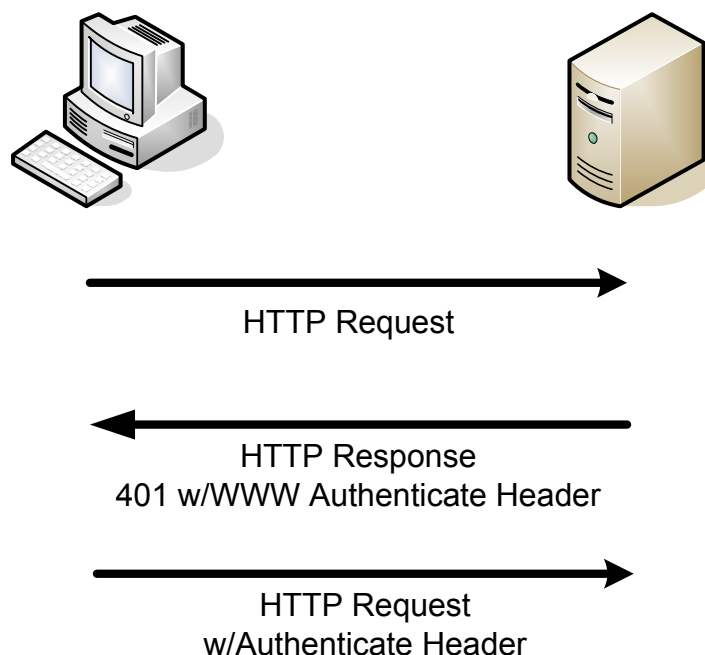
Background

By default, many web servers are configured to receive access requests via Basic Access Authentication. Over this mechanism, a user's credentials are generally sent in cleartext and can easily be intercepted by a hacker. In general, this mechanism is acceptable for only the least sensitive forms of data, such as content from a publicly-accessible store or repository. Secure Sockets Layer (SSL) authentication (discussed in HOL 111) is a vast improvement over Basic Access Authentication but isn't desirable in all circumstances due to the following issues:

1. All communication performed over SSL is encrypted, meaning additional bandwidth is consumed and both the host and client must dedicate processing time to encrypting/decrypting packets.
2. Proper SSL authentication requires the use of a Certificate Authority (CA) signed certificate, with the certificate being installed on the server to provide a means by which clients can validate the server is legitimate. Obtaining such a certificate has a cost and maintenance overhead associated.

Digest Access Authentication is seen as a "happy medium" between Basic Access Authentication and SSL in that the initial identity check is encrypted but all packets beyond that are not. Digest Access Authentication uses a challenge/response system in which both the server and client must agree on a particular sequence of events where if a single portion of the exchange is missed or dropped, the request must be restarted from the beginning. The challenge sent by the server contains a portion of variable-size/content data. The client then uses a key generated from the submitted password to encrypt the challenge. This forms the response back to the server. The server can then take the encrypted password and look that up against an Active Directory (in this case, a Padarn configuration file) and apply a known encryption routine against the looked up password (MD5 is the encryption

method supported by Padarn) and verify the two encrypted passwords match. If there is a match, the response sent back is the actual requested content. In this lab, the response sent back will be the webpage requested by the client.



Here is a bit more detail as to how this process takes place:

1. Client requests access to a Padarn resource (generally an ASPX file, such as the default website page)
2. Padarn recognizes the request as needing a protected resource. Padarn recognizes the client has not recently requested this resource and so it's treated as a first-time request. A header is sent back to the client with the following particulars:
 - a. The realm (domains are not supported)
 - b. A timestamp, a nonce (see below), and an HTML tag corresponding to the resource requested – all encrypted using a private key. This is the security context (also known as the opaque)
 - c. A value indicating the type of encryption supported

This information is sent back to the client in a 401 message code, which appears as an unauthorized access error. See Exercise 2 below to see how this content appears through a network packet analyzer.

3. Next, the client must respond with the following pieces of information:
 - a. The user name and password combined with the original nonce
 - b. The original nonce along with the number of times the nonce has been used (if this count is inaccurate, a denial of service attack can be prevented)
 - c. The original resource path requested by the client
 - d. A value indicating the type of encryption supported

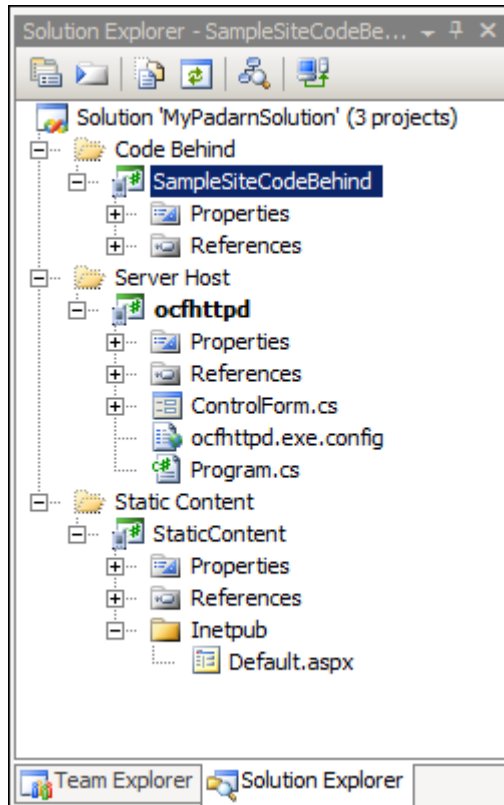
4. Upon receiving this response, the server checks:
 - a. Whether this challenge response was a challenge against the Padarn server and not another server
 - b. The contents of the nonce-combined credentials and URI are all valid
 - c. The nonce count is accurate and doesn't indicate a challenge had come back with the same nonce an earlier time

If all checks pass, then the client is allowed to access the resource. Otherwise, the client will be sent an unauthorized access challenge and the process continues.

If all checks pass, subsequent attempts by the client to gain access to the *same* resource can re-use the security context and bypass the initial 401 unauthorized access challenge. Implementations of when the security context grows stale differ by platform.

Exercise 1: Configuring a basic Padarn web resource

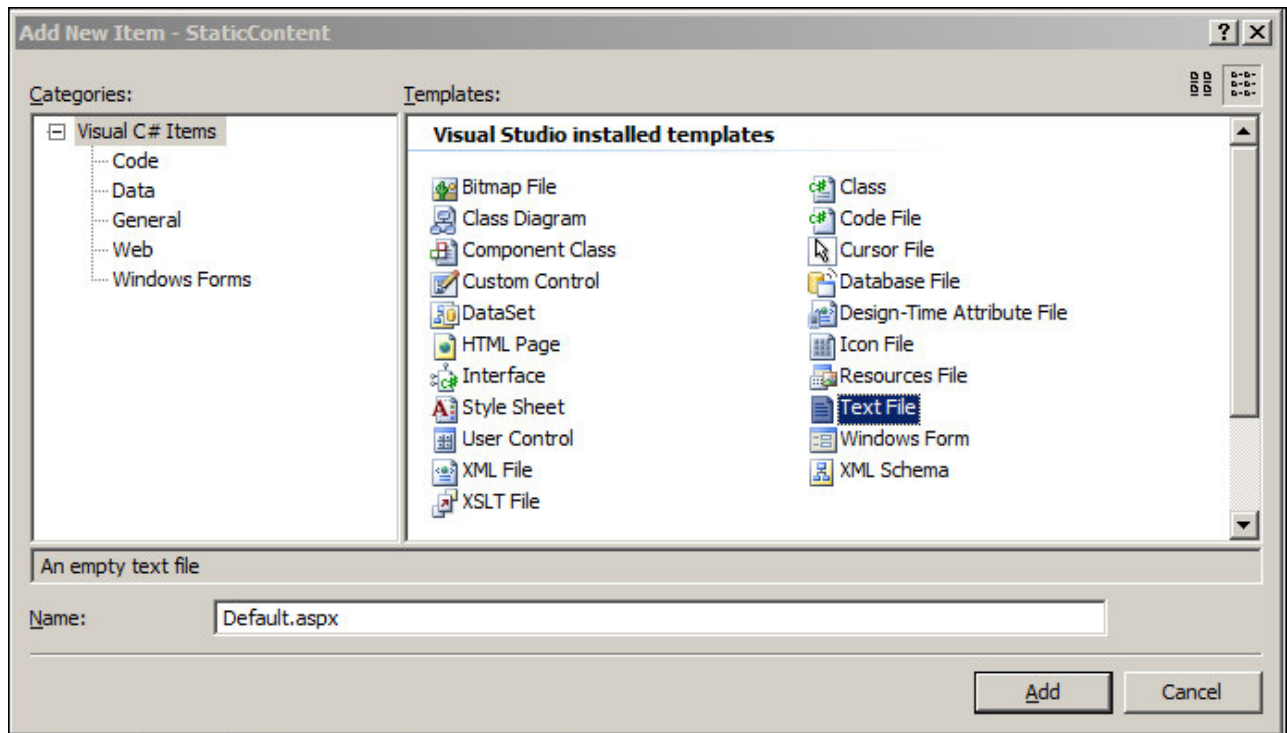
In this exercise, we will configure Padarn to serve a single Active Server Page (ASP) and we'll create some basic HTML as the test harness. To begin, please open the Visual Studio 2008 solution entitled "MyPadarnSolution.sln". Here is how the Solution Explorer displays the contents of this solution:



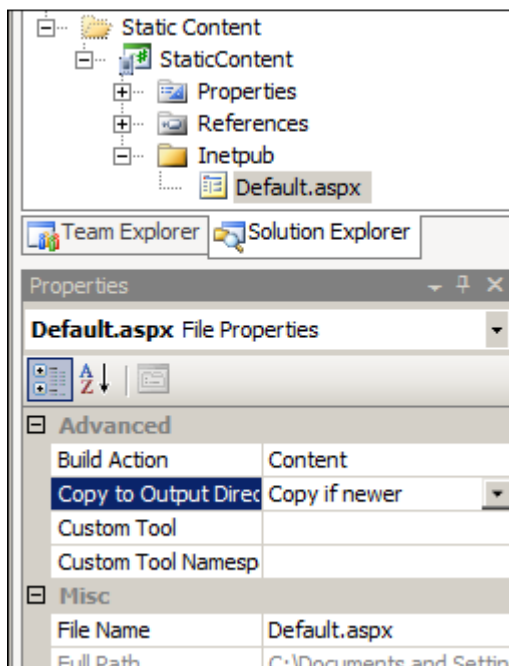
The two projects listed are a sample Padarn web server launch utility (Smart Device project) and an empty project to hold static content. We will now add a single Active Service Page (ASP) to the StaticContent project as our protected (and default) webpage for the Padarn instance.

To create the ASPX file:

1. In Visual Studio's **Solution Explorer** Pane, right-click on the **Inetpub** folder in the **StaticContent** project and select **Add -> New Item** to display the **Add New Item** dialog.
2. From the list of **Visual Studio Templates**, select **Text File**
3. In the **Name** textbox enter "Default.aspx"



4. Click **Add** to add the new file to the project
5. In Visual Studio's Solution Explorer Pane, click on the newly-created Default.aspx file
6. In the Properties pane, set the Build Action for Default.aspx to "Content"
7. In the Properties pane, set the Copy to Output Directory property for Default.aspx to "Copy if Newer"



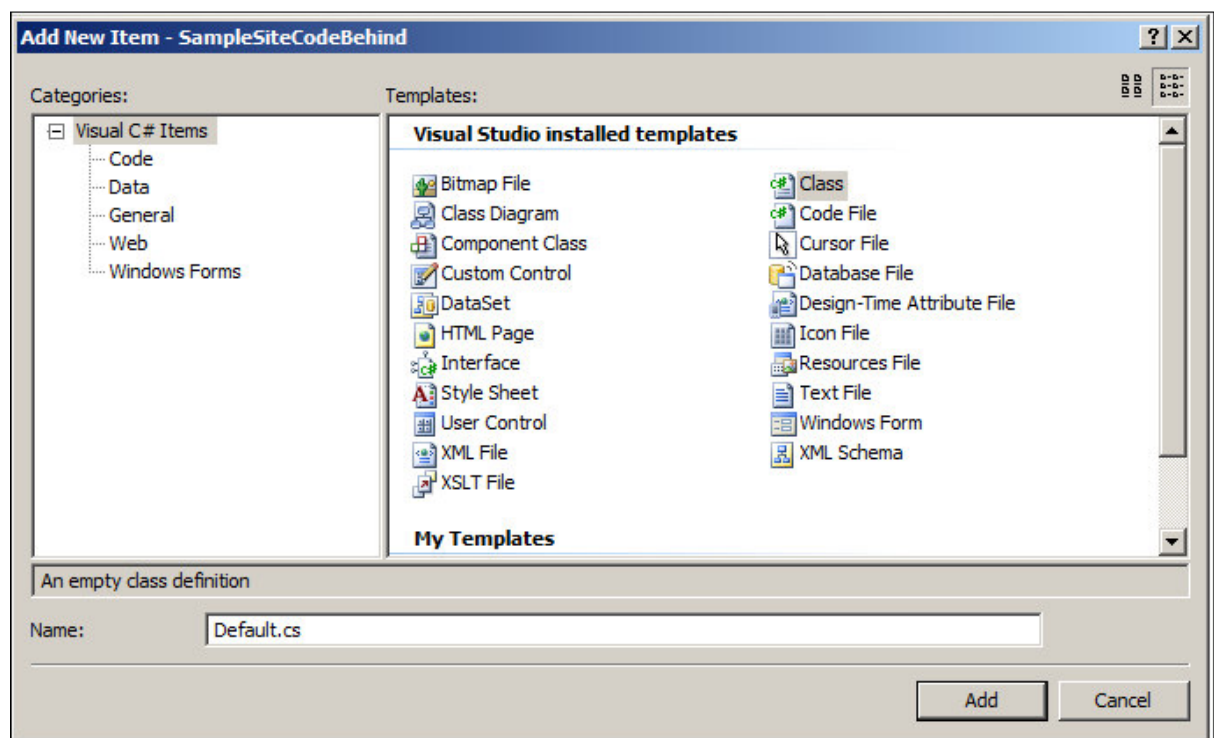
8. Paste the following code into the newly-created Default.aspx file:

```
<%@ Page CodeBehind="SampleSite.dll" Inherits="SampleSite.Default" %>
```

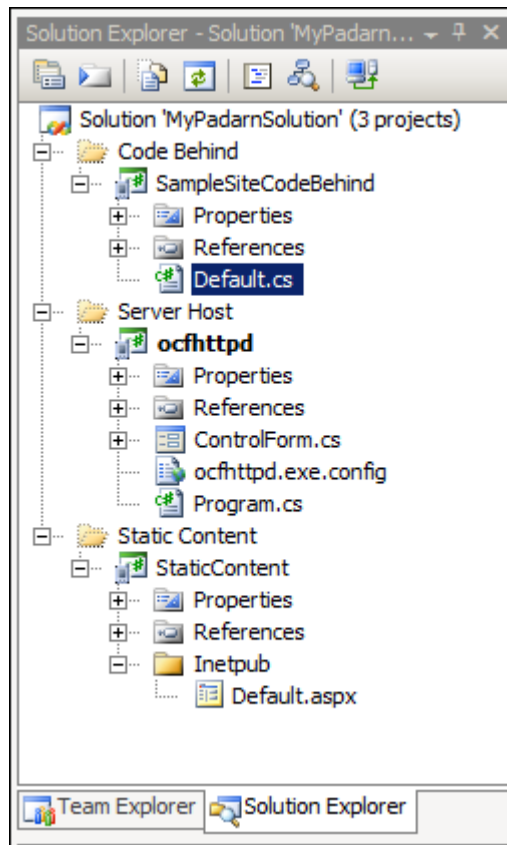
Next, you will use Visual Studio to create classes containing the code-behind logic for the sample startup webpage. The Padarn server will create an instance of the main class and run the Page_Load method of the class when a client browser (or Net client) accesses the Default.aspx page.

To create the main code-behind class:

1. In Visual Studio's Solution Explorer pane, right-click on the SampleSiteCodeBehind project and select **Add -> New Item** to display the Add New Item dialog.
2. From the list of Visual Studio Installed Templates select **Class**.
3. In the Name textbox, enter "Default.cs".



4. Click **Add** to add the new file to the project
5. The solution should now look the following:



At this point, we can define a very simple “Padarn Hello World” output. To do that, add the following code to Default.cs:

```
using System;
using OpenNETCF.Web.UI;
using OpenNETCF.Web.Html;

namespace SampleSite
{
    public class Default : Page
    {
        protected override void Page_Load(object sender, EventArgs e)
        {
            // create the document
            Document doc = new Document();

            // add a header to the document
            doc.Head = new DocumentHead("OpenNETCF Padarn Web Server");

            #region --- other site links ---
            Div menuContainer = new Div();
            menuContainer.ClassName = "centeredContainer";

            Div menuItemContainer = new Div();
            menuItemContainer.ClassName = "siteMenu";

            Paragraph paragraph = new Paragraph();
            paragraph.Elements.Add(new FormattedText("Hello World Padarn!", TextFormat.Bold,
                "20"));
            paragraph.Elements.Add(new LineBreak());
            paragraph.Elements.Add(new FormattedText("If you are viewing this page, your " +
                "credentials were accepted (though this connection is not encrypted)",
                TextFormat.Bold));

            menuItemContainer.Elements.Add(paragraph);
            menuContainer.Elements.Add(menuItemContainer);
            doc.Body.Elements.Add(menuContainer);
            #endregion

            // send the document html to the Response object
            Response.Write(doc.OuterHtml);

            // flush
            Response.Flush();
        }
    }
}
```

Exercise 2: Configuring Padarn to use Digest Access Authentication

The interesting part of this lab is also the simplest to configure, believe it or not! 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="80"
  MaxConnections="20"
  DocumentRoot="\Windows\Inetpub\"
  Logging="true"
  LogFolder="\Temp\Logs"
  LogExtensions="aspx;html;htm;zip"
  BrowserDefinitions="\Windows\Inetpub\config\browsers"
  UseSsl="false"
  CertificateName="\Windows\certificate\server.pfx"
  CertificatePassword="padarn"
>
<DefaultDocuments>
  <Document>default.aspx</Document>
</DefaultDocuments>
```

Verify the Authentication mode of Padarn is set to Digest in the `Authentication` section:

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

1. Make sure the authentication mode is set to Digest. (Basic is supported as a value as well).
Note: this is not the area where SSL will be configured. See HOL 111 for more details.
2. Make sure authentication is enabled
3. Enter an appropriate user name and password value pair. Domain is not presently supported (as of v1.1.78)

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 3: Verify Digest Authentication

In order to validate that our authentication scheme is working properly, 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.

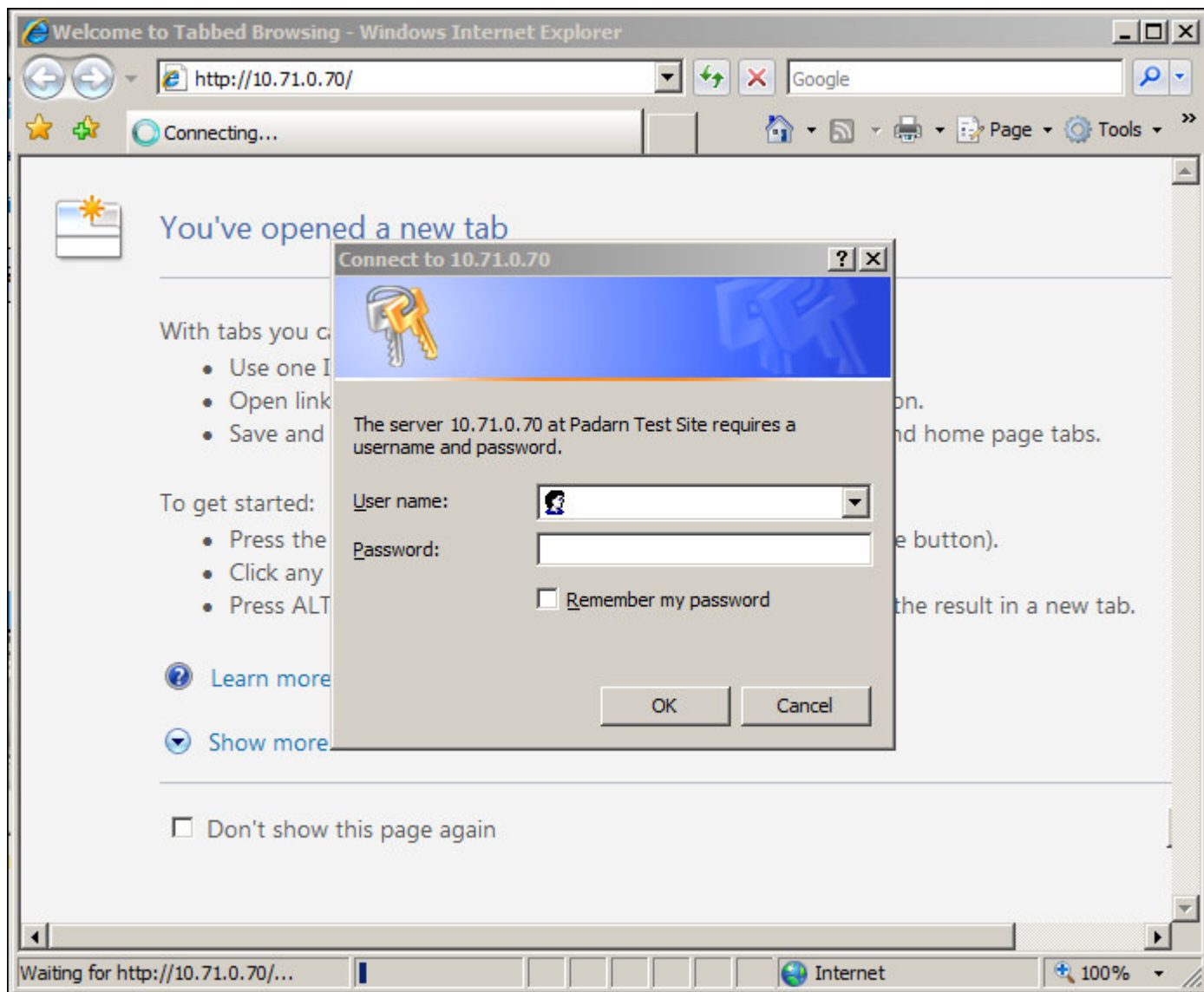
1. Initiate Padarn through the MyPadarnSolution Visual Studio 2008 solution. Right-click on the ocfshttpd project and select **Debug -> Start Without Debugging** (or press Ctrl + F5). For this demonstration, we will be using the Windows Mobile 5.0 Pocket PC R2 Emulator.



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



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. The first thing we see when attempting to visit the base URL of the Padarn server is a credential challenge (Internet Explorer interprets this challenge and provides a human-usable set of prompts to collect this information):



Notice the realm text displays just as we defined it in the configuration file.

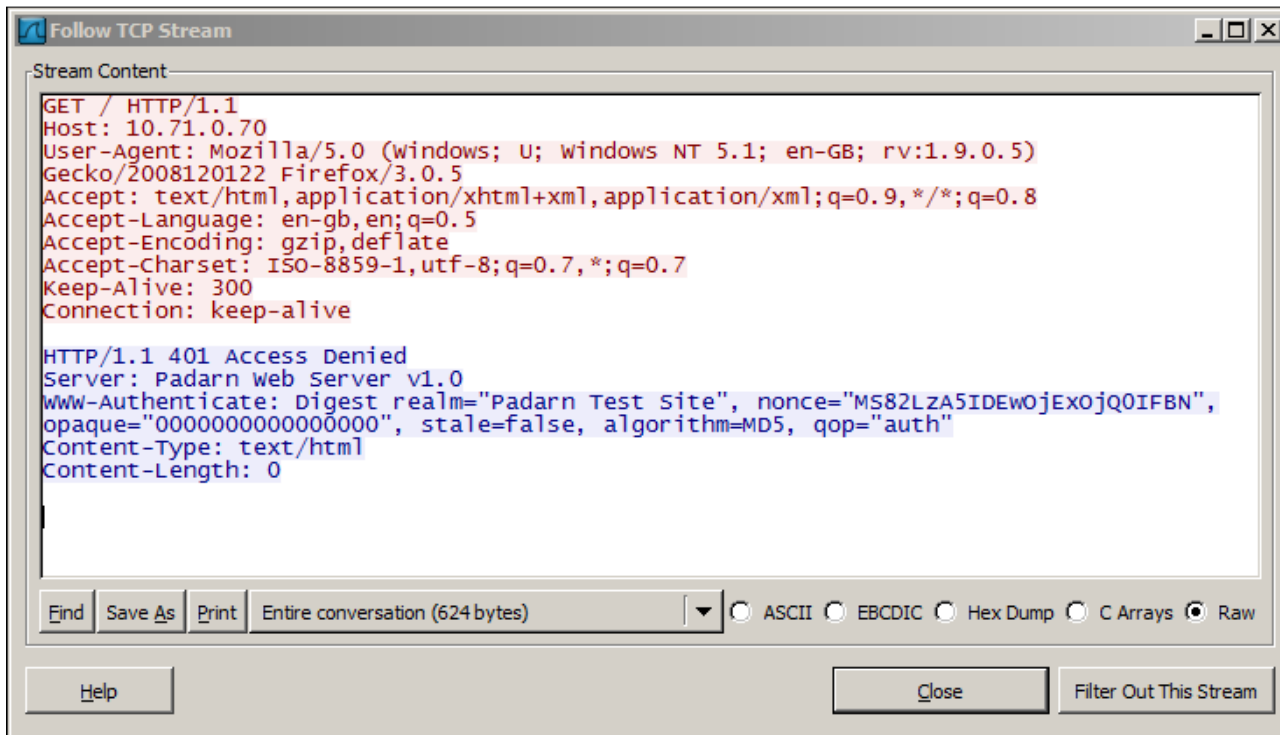
Should we enter valid credentials, we can use a network sniffer tool such as Wireshark ([information here](#)) to determine all headers exchanged between Padarn and the desktop instance. Here is a brief snapshot of what we would expect to see on the server:

1290	155.627539	10.71.0.76	10.71.0.70	HTTP	GET / HTTP/1.1
1298	158.750182	10.71.0.70	10.71.0.76	HTTP	HTTP/1.1 401 Access Denied
1314	164.198114	10.71.0.76	10.71.0.70	HTTP	GET / HTTP/1.1
1327	167.087671	10.71.0.70	10.71.0.76	HTTP	HTTP/1.1 200 OK (text/html)

You can trace the request – challenge – response – acknowledge sequence from the beginning of the HOL.

1. A request is made by client (10.71.0.76) to fetch the base webpage of the server (10.71.0.70).

- The server denies access but sends the client security context information so the client can return encrypted credentials.

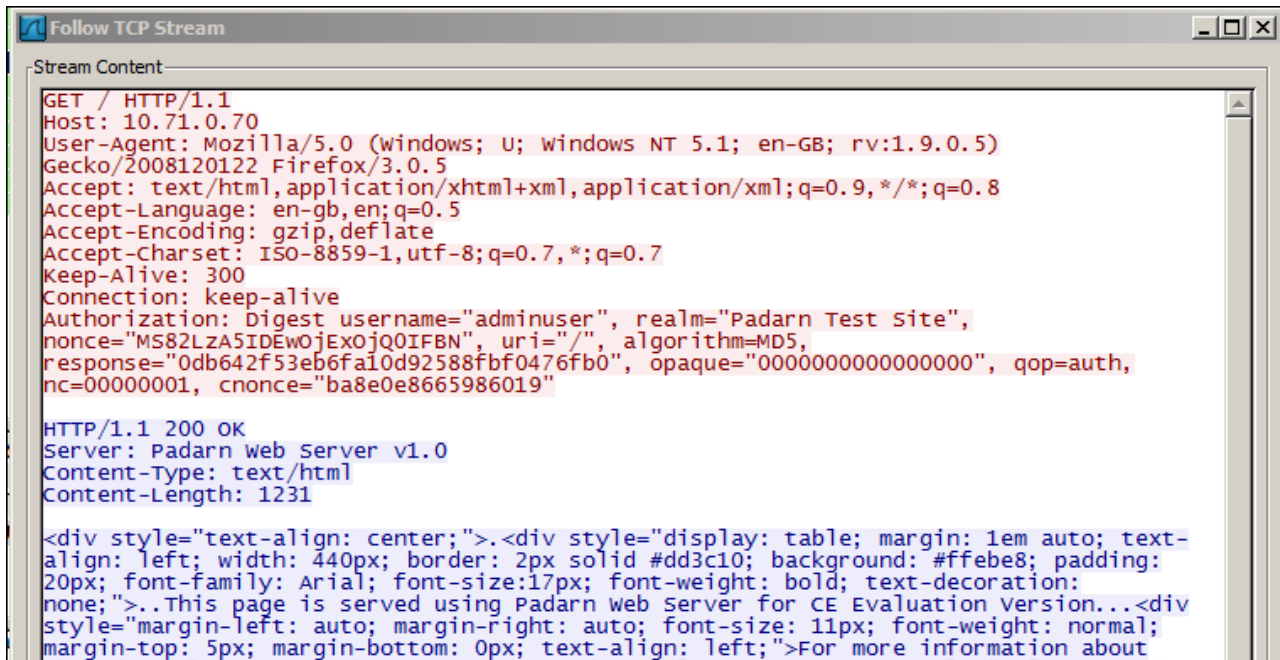


```

GET / HTTP/1.1
Host: 10.71.0.70
User-Agent: Mozilla/5.0 (windows; U; windows NT 5.1; en-GB; rv:1.9.0.5)
Gecko/2008120122 Firefox/3.0.5
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive

HTTP/1.1 401 Access Denied
Server: Padarn web Server v1.0
WWW-Authenticate: Digest realm="Padarn Test Site", nonce="MS82LZA5IDewOjExOjQ0IFBN",
opaque="0000000000000000", stale=false, algorithm=MD5, qop="auth"
Content-Type: text/html
Content-Length: 0
  
```

- The client submits all requested details from the server's challenge



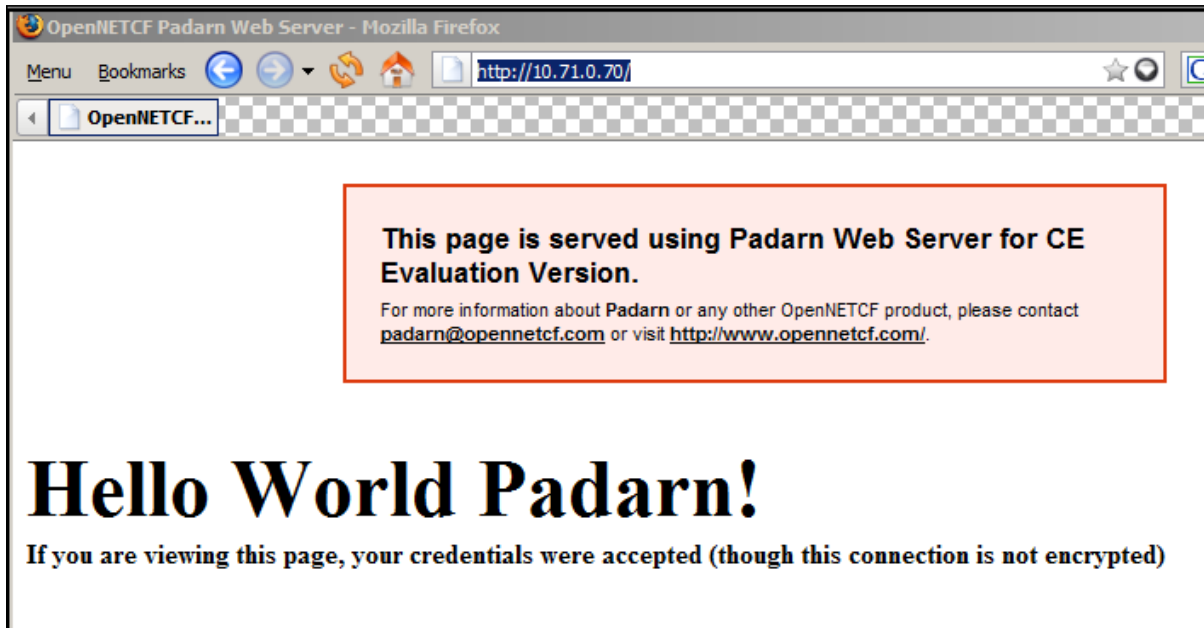
```

GET / HTTP/1.1
Host: 10.71.0.70
User-Agent: Mozilla/5.0 (windows; U; windows NT 5.1; en-GB; rv:1.9.0.5)
Gecko/2008120122 Firefox/3.0.5
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Authorization: Digest username="adminuser", realm="Padarn Test Site",
nonce="MS82LZA5IDewOjExOjQ0IFBN", uri="/", algorithm=MD5,
response="0db642f53eb6fa10d92588fbf0476fb0", opaque="0000000000000000", qop=auth,
nc=00000001, cnonce="ba8e0e8665986019"

HTTP/1.1 200 OK
Server: Padarn web Server v1.0
Content-Type: text/html
Content-Length: 1231

<div style="text-align: center;"><div style="display: table; margin: 1em auto; text-
align: left; width: 440px; border: 2px solid #dd3c10; background: #ffebe8; padding:
20px; font-family: Arial; font-size: 17px; font-weight: bold; text-decoration:
none;">..This page is served using Padarn web Server for CE Evaluation version...<div
style="margin-left: auto; margin-right: auto; font-size: 11px; font-weight: normal;
margin-top: 5px; margin-bottom: 0px; text-align: left;">For more information about
  
```

- The server provides the client with the requested data:



This level of detail is given to provide the user with a deep dive into how Padarn implements Digest Access Authentication. Rest assured – your credentials are safe from prying eyes!

Hands-on Lab Summary

It is important in this day and age to use reasonable security measures for all computing devices, no matter how small that device might be. While a Windows CE handheld might not have a multi-core processor and several gigabytes of RAM, it has an architecture that makes running Win32 code quite easy. Having a server sitting on your network, albeit a lightweight web server, still opens you up for credential stealing.

If you are unable or unwilling to encrypt all traffic over SSL, then Digest Access Authentication is a straightforward mechanism that is widely supported and has no real impact to clients.

In this lab you:

- ✓ Re-used an existing your Padarn website solution
- ✓ Made a simple change to the Padarn ocfhttpd.exe.config to enable Digest Access Authentication
- ✓ Made a simple “Hello World” style test landing page for verification of a valid authentication
- ✓ Stepped through the 4-part Digest “handshake” within a network packet analyzer to verify encryption of credentials

In HOL 111, we will investigate the necessary tools for protecting all data transferred between a client and a Padarn instance via Secure Sockets Layer (SSL).