From: https://stackoverflow.com/questions/11403333/httplistener-with-https-support/33905011#33905011

Here are the steps, in detail, that I followed to set up a stand-alone server on Windows, using OpenSSL to create the self-signed certificate for a C# `HTTPListener` application. It includes plenty of links, in case you want to do further research.

1. Create a stand-alone server in .NET via `HttpListener`:

```
var prefixes = {"http://localhost:8080/app/root",
"https://localhost:8443/app/root"};
var listener = new HttpListener();
foreach (string s in prefixes)
    listener.Prefixes.Add(s);
listener.Start();
```

7. Create self-signed certificate:*
    1. `openssl req -x509 -newkey rsa:2048 -keyout key.pem -out cert.pem -days 365`, which will prompt you for the value of each of the certificate's fields on the command line. For the common name, type the domain name (e.g. `localhost`)
    2. `openssl pkcs12 -inkey bob_key.pem -in bob_cert.cert -export -out bob_pfx.pfx`, so that it can be imported with its key on the target machine.

    *For an alternative using `makecert`, see Walter's own answer.

8. Open Certificate Manager for the Local Machine. When you run `certmgr.msc`, it opens the Certificate Manager for the *current user*, which is not what we want here. Instead:
    1. From an administrative command prompt on the target machine, run `mmc`
    2. Press `Ctrl + M`, or Click `File > Add/Remove Snap-in`
    3. Choose `Certificates`, and click `Add >`
    4. In the dialog that appears, Choose `Computer Account`, and click `Next`
    5. Choose `Local Computer`. Click `Finish`, then `Okay`
9. Import the certificate (`pfx`) into the Windows Certificate Store on the target machine
    1. In the `mmc` window previously opened, drill down to `Certificates (Local Computer) > Personal`
    2. Right-click on `Personal`, then click on `All Tasks -> Import...`
    3. In the 2nd screen of the dialog that appears, find and import your certificate. You'll have to change the file-type filter to `Personal Information Exchange` or `All Files` in order to find it
    4. On the next screen, enter the password you chose in step 2.1, and pay close attention to the first check box. This determines how securely your certificate is stored, and also how convenient it is to use
    5. On the last screen, choose `Place all certificates in the following store`. Verify that it says `Personal`, then click `Finish`
    6. Repeat the import procedure above for the `Trusted Root Certification Authorities` certificates section.
10. Create the port associations for your application. On Windows Vista and later, use `netsh`, as I did. (For Windows XP and earlier, use `httpcfg`)

- From the administrative command line, type the following to set up the SSL binding* to your app, and the appropriate port. **NB:** This command is easy to get wrong, because (in PowerShell) the braces need to be escaped. The following PowerShell command will work:

```
netsh http add sslcert ipport=0.0.0.0:8443 `
    certhash=110000000000003ed9cd0c315bbb6dc1c08da5e6 `
    appid=`{00112233-4455-6677-8899-AABBCCDDEEFF`}
```

For `cmd.exe`, the following should be used instead:

```
netsh http add sslcert ipport=0.0.0.0:8443
certhash=110000000000003ed9cd0c315bbb6dc1c08da5e6
appid={00112233-4455-6677-8899-AABBCCDDEEFF}
```

  - The `ipport` parameter will cause the SSL certificate to bind to the port `8443` on every network interface; to bind to a specific interface (only), choose the IP address associated with that network interface.
  - The `certhash` is simply the certificate thumbprint, with spaces removed
  - The `appid` is the GUID stored in the Assembly Info of your application. (Sidenote: The `netsh` mechanism is evidently a COM interface, judging from this question and its answers)

* Microsoft has redirected the *SSL Binding* link from here to there.

11. Start up your web-server, and you're good to go!